18th International Conference on Accelerator and Large Experimental Physics Control Systems

Tracks

1. General
Project Status Reports
Control System Upgrades
Device Control and Integrating Diverse Systems
Experiment Control

2. Software
Software Technology Evolution
User Interfaces and User eXperience (UX)
Data Management
Data Analytics

3. Hardware
Hardware Technology
Timing Systems, Synchronization and Real-Time Applications
Control System Infrastructure

4. Subsystems
Human Aspects, Collaborations, Management
Functional Safety Systems for Machine Protection, Personnel Safety
Feedback Control, Machine Tuning and Optimization

ICALEPCS 2021 Virtual

14-22 October 2021

Local Organizing Committee

Yongbin Leng, Conference Chair
Yingbing Yan, Program Chair
Zhichu Chen, Proceedings Chief Editor
Pengxiang Yu, Conference Secretary
Min Li, Finance Assistant
Qingwen Xiao, IT Coordinator
Xiaomin Liu, Oral Manager
Danping Bai, Poster Manager
Huihui Lv, Industrial Exhibition Manager
Heyun Wang, Web & Whova Manager

PROCEEDINGS

Hosted by :
Shanghai Advanced Research Institute
Chinese Academy of Sciences

Foreword to the ICALEPCS 2021 Proceedings

The 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021) was held from the 14th October to the 22nd October in virtual format due to COVID-19 pandemic. It was hosted by Shanghai Advanced Research Institute. The conference was endorsed by the Chinese Academy of Sciences (CAS), the National Natural Science Foundation of China (NSFC), the European Physical Society (EPS), the American Physical Society (APS), and the Shanghai Nuclear Society. The conference's Local Organizing Committee was advised by two committees made up of members from the three regions – the International Scientific Advisory Committee (ISAC) and the Program Committee (PC). The ISAC was comprised of 38 members and chaired by Kazuro Furukawa from KEK. The PC was made up of 40 members and chaired by Yingbing Yan from SARI.

In total 552 participants registered. Of these, 39 were students. CERN was the most represented institute followed by SARI. The breakdown of attendees by region was as follows: 113 from Asia and Oceania, 371 from Europe, Middle East and Africa, and 68 from North and South America.

A total of 290 abstracts were accepted. Of these, 84 were selected as contributed orals and 206 were poster presentations. For the proceedings, 218 papers were uploaded and processed to be included.

ICALEPCS 2021 put a strong emphasis on having quality and diverse keynote speakers. Zhentang Zhao, from SARI, spoke on Highlights of Accelerator R&D at Shanghai Light Sources Cluster. Lei Ding, from SITP, spoke on Infrared Remote Sensors for Chinese Meteorological Satellites. Juan Yin, from USTC, spoke on Large-scale Quantum Communication Network. Dejun Ning, from SARI, spoke on Digital Transformation of Urban Economy and Case Study. And, Chen Lin, from Peking University, spoke on Laser Ion Accelerator and Its Applications.

The conference themes were chosen by the ISAC. The scientific program was drawn up by the PC. Both committees did an excellent job choosing topics and talks and encouraging experts to present their work. Parallel sessions were necessary, as have been in the past. Out of a total of 27 sessions, 20 were parallel.

There were 5 pre-conference workshops held on the Thursday (14th October) and Friday (15th October) before the conference. The workshop themes were: Linux Distribution and Ecosystem Evolution in Control Systems, TANGO, Data Science and Machine Learning, PLC Based Control Systems, and Motion Control. The attendance of the pre-conference workshops was very high. More than 450 people attended the workshops.

Two awards were given at ICALEPCS 2021, the ICALEPCS Lifetime Achievement Award (LAA) and the ICALEPCS Leadership in Mentoring Award (LMA). The ICALEPCS 2021 LAA award was presented to Andy Götz and Roland Müller. As members of ISAC and IEC, through their organizational skills and vision, as well as technical excellence and tireless activity, they have been the key to the success of two

decades of ICALEPCS conferences. The ICALEPCS 2021 LMA award was presented to Songqiang Liu. With his dedication to training and mentoring over the decades, he successfully grew highly qualified controls engineers, and contributed to the international accelerator community and EPICS collaboration.

The ISAC accepted the bid for ICALEPCS 2025 to be held in Chicago, USA, hosted by Advanced Photon Source (APS)/Argonne National Laboratory (ANL).

As chairs we would like to thank everyone who contributed to make ICALEPCS 2021 a successful and memorable event. The conference was the result of many people's work and support both financial and moral. It all started in 2017 in Barcelona, when the ISAC of ICALEPCS 2017 accepted the SSRF bid to host ICALEPCS 2021 in Shanghai. It continued with ICALEPCS 2019 who sponsored ICALEPCS 2021 with the earnings that remained from ICALEPCS 2019.

We thank all the sponsors and exhibitors who helped finance the conference. They were Beckhoff, Cosylab, D-TACQ Solutions, Instrumentation Technologies, nVent, Plant & Mill Supplies, RadiaSoft LLC, Technosystem and Teledyne SP Devices.

The ISAC chair and committee did a great job advising the LOC during the 2 years leading up to the conference. The PC did an excellent job shaping the scientific program. The editors did an excellent job editing the many abstracts and papers to ensure that the papers were of high quality. The delegates made the conference a success by participating with high quality talks and questions. A big thanks to SARI, the host institute, and all the colleagues who helped and provided support for ICALEPCS 2021.

We want to thank our JACoW team: Zhichu Chen, Volker RW Schaa, Raphael Müller, Akihiro Shirakawa, Lu Li, and the SPMS regional support from Takashi Kosuge.

Finally, we would like to thank all the LOC members who worked very hard to ensure ICALEPCS 2021 was a success!

We look forward to seeing everyone in Cape Town in 2023 and in Chicago in 2025!

Conference Chair: Yongbin Leng
Program Chair: Yingbing Yan

International Scientific Advisory Committee (ISAC)

Reinhard Bacher		DESY	Europe, Middle East & Africa
Matthew Bickley		JLAB	Americas
Enrique Blanco		CERN	Europe, Middle East & Africa
Sandor Brockhausen		XFEL	Europe, Middle East & Africa
Alain Buteau		SOLEIL	Europe, Middle East & Africa
Yung-Sen Cheng		NSRRC	Asia & Pacific
Gianluca Chiozzi		ESO	Europe, Middle East & Africa
Paul Chu		IHEP	Asia & Pacific
Scott Cogan		FRIB	Americas
Mikhail Fedorov		LLNL	Americas
Richard Farnsworth		BNL	Americas
Daniel Flath		SLAC	Americas
Kazuro Furukawa	**Chair**	KEK	Asia & Pacific
Vincent Hardion		MAXIV	Europe, Middle East & Africa
Nick Hauser		ANSTO	Australia
Markus Janoush		PSI	Europe, Middle East & Africa
Nicolas Janvier		ESRF	Europe, Middle East & Africa
Norihiko Kamikubota		KEK	Asia & Pacific
Misaki Komiyama		RIKEN	Asia & Pacific
Timo Korhonen		ESS	Europe, Middle East & Africa
Ge Lei		IHEP	Asia & Pacific
Yongbin Leng		SARI	Asia & Pacific
Marco Lonza		ELETTRA	Europe, Middle East & Africa
Oscar Matilla		ALBA	Europe, Middle East & Africa
Roland Müller		HZB	Europe, Middle East & Africa
Seth Nemesure		BNL	Americas
Dennis Nicklaus		Fermilab	Americas
Eric Norum		Berkeley	Americas
Stephane Perez		CEA	Europe, Middle East & Africa
Martin Pieck		LANL	Americas
Austen Rose		DLS	Europe, Middle East & Africa
Lucas Russo		LNLS	Americas
Choji Saji		SPRING-8	Asia & Pacific
Javier Serrano		CERN	Europe, Middle East & Africa
Guobao Shen		Argonne	Americas
Gabriele Vajente		LIGO	Americas
Johan Venter		SKA	Europe, Middle East & Africa
Karen White		SNS	Americas
Yingbing Yan		SSRF	Asia & Pacific

International Executive Committee (IEC)

Kazuro Furukawa	**Chair**	KEK	Asia & Pacific
Seth Nemesure		BNL	Americas
Johan Venter		SKA	Europe, Middle East & Africa
Karen White		SNS	Americas
Yingbing Yan		SSRF	Asia & Pacific

Program Committee (PC)

Reinhard Bacher		DESY	Europe, Middle East & Africa
Matthew Bickley		JLAB	Americas
Enrique Blanco		CERN	Europe, Middle East & Africa
Sandor Brockhausen		XFEL	Europe, Middle East & Africa
Alain Buteau		SOLEIL	Europe, Middle East & Africa
Yung-Sen Cheng		NSRRC	Asia & Pacific
Gianluca Chiozzi		ESO	Europe, Middle East & Africa
Paul Chu		IHEP	Asia & Pacific
Scott Cogan		FRIB	Americas
Mikhail Fedorov		LLNL	Americas
Daniel Flath		SLAC	Americas
Kazuro Furokawa		KEK	Asia & Pacific
Manuel Gonzalez-Berges		CERN	Europe, Middle East & Africa
Vincent Hardion		MAXIV	Europe, Middle East & Africa
Eugenia Hatziangeli		CERN	Europe, Middle East & Africa
Nick Hauser		ANSTO	Australia
Marcus Janoush		PSI	Europe, Middle East & Africa
Nicolas Janvier		ESRF	Europe, Middle East & Africa
Norihiko Kamikubota		KEK	Asia & Pacific
Misaki Komiyama		RIKEN	Asia & Pacific
Timo Korhonen		ESS	Europe, Middle East & Africa
Ge Lei		IHEP	Asia & Pacific
Yongbin Leng		SARI	Asia & Pacific
Marco Lonza		ELETTRA	Europe, Middle East & Africa
Oscar Matilla		ALBA	Europe, Middle East & Africa
Roland Müller		HZB	Europe, Middle East & Africa
Seth Nemesure		BNL	Americas
Dennis Nicklaus		Fermilab	Americas
Takashi Obina		KEK	Asia & Pacific
Stephane Perez		CEA	Europe, Middle East & Africa
Martin Pieck		LANL	Americas
Austen Rose		DLS	Europe, Middle East & Africa
Lucas Russo		LNLS	Americas
Choji Saji		SPRING-8	Asia & Pacific
Javier Serrano		CERN	Europe, Middle East & Africa
Guobao Shen		Argonne	Americas
Gabriele Vajente		LIGO	Americas
Johan Venter		SKA	Europe, Middle East & Africa
Karen White		SNS	Americas
Yingbing Yan	**Chair**	SSRF	Asia & Pacific

Local Organizing Committee (LOC)

Yongbin Leng	**Chair**	Conference Chair
Yingbing Yan		Program Chair
Zhichu Chen		Proceedings Chief Editor
Pengxiang Yu		Conference Secretary
Min Li		Finance Assistant
Qingwen Xiao		IT Coordinator
Xiaomin Liu		Oral Manager
Danping Bai		Poster Manager
Huihui Lv		Industrial Exhibition Manager
Heyun Wang		Web & Whova Manager

JACoW Editorial Team (JET)

Zhichu Chen	**Editor in Chief**	SARI
Volker RW Schaa	**Technical Editor**	GSI
Akihiro Shirakawa		KEK
Raphael Müller		GSI
Lu Li		IMP

Contents

Preface i
Foreword . . . iii
Committees . . . v
Contents . . . ix

Papers 1
MOAL01 – Maturity of the MAX IV Laboratory in Operation and Phase II Development . . . 1
MOAL02 – Status of the National Ignition Facility (NIF) Integrated Computer Control and Information Systems . . . 9
MOAL03 – From SKA to SKAO: Early Progress in the SKAO Construction . . . 14
MOAR01 – Modernizing the SNS Control System . . . 21
MOAR02 – Modernizing Digital Video Systems at the National Ignition Facility (NIF): Success Stories, Open Challenges and Future Directions . . . 26
MOAR03 – LOFAR2.0: Station Control Upgrade . . . 31
MOBL01 – The ELT Control System: Recent Developments . . . 37
MOBL02 – Real-Time Framework for ITER Control Systems . . . 45
MOBL03 – Machine Learning Platform: Deploying and Managing Models in the CERN Control System . . . 50
MOBL04 – Karabo Data Logging: InfluxDB Backend and Grafana UI . . . 56
MOBL05 – Photon Science Controls: A Flexible and Distributed LabVIEW Framework for Laser Systems 62
MOBR01 – ROMULUSLib: An Autonomous, TCP/IP-Based, Multi-Architecture C Networking Library for DAQ and Control Applications . . . 69
MOBR02 – Control, Readout and Monitoring for the Medium-Sized Telesopes in the Cherenkov Telescope Array . . . 77
MOBR03 – Hexapod Control System Development Towards Arbitrary Trajectories Scans at Sirius/LNLS 84
MOBR04 – Generic Data Acquisition Control System Stack on the MTCA Platform . . . 90
MOPV001 – Status of the SARAF-Phase2 Control System . . . 93
MOPV002 – CENBG Control System and Specific Instrumentation Developments for SPIRAL2-DESIR Setups . . . 98
MOPV003 – Laser Megajoule Facility Operating Software Overview . . . 104
MOPV005 – Towards a New Control System for PETRA IV . . . 108
MOPV006 – The New Small Wheel Low Voltage Power Supply DCS for the ATLAS Experiment . . . 111
MOPV009 – The HV DCS System for the New Small Wheel Upgrade of the ATLAS Experiment . . . 115
MOPV010 – Working under Pandemic Conditions: Contact Tracing Meets Technology . . . 121
MOPV011 – The Inclusion of White Rabbit into the Global Industry Standard IEEE 1588 . . . 126
MOPV012 – The ESRF-EBS Simulator: A Commissioning Booster . . . 132
MOPV013 – A Dynamic Beam Scheduling System for the FAIR Accelerator Facility . . . 138
MOPV014 – Upgrade of the NewSUBARU Control System . . . 143
MOPV015 – Control System of the SRILAC Project at RIBF . . . 147
MOPV016 – Design and Implement of Web Based SCADA System for HUST Field-Reversed Configuration Device . . . 153
MOPV017 – CERN SCADA Systems 2020 Large Upgrade Campaign Retrospective . . . 156
MOPV018 – Linac-200 Gun Control System: Status and Plans . . . 161
MOPV019 – PVEcho: Design of a Vista/EPICS Bridge for the ISIS Control System Transition . . . 164
MOPV020 – Digitisation of the Analogue Waveform System at ISIS . . . 169
MOPV021 – Upgrading the National Ignition Facility's (NIF) Integrated Computer Control System to Support Optical Thompson Scattering (OTS) Diagnostic . . . 173
MOPV024 – vscode-epics, a VSCode Module to Enlighten Your EPICS Code . . . 179
MOPV025 – TangoGraphQL: A GraphQL Binding for Tango Control System Web-Based Applications . . . 181
MOPV026 – Integrating OPC UA Devices in EPICS . . . 184
MOPV027 – The Evolution of the DOOCS C++ Code Base . . . 188
MOPV030 – Application of EPICS Software in Linear Accelerator . . . 193
MOPV031 – The Deployment Technology of EPICS Application Software Based on Docker . . . 197
MOPV032 – Design of a Component-Oriented Distributed Data Integration Model . . . 202
MOPV033 – Web Client for Panic Alarms Management System . . . 206
MOPV034 – Migration of Tango Controls Source Code Repositories . . . 209

MOPV035 – Development of Alarm and Monitoring System Using Smartphone . . . 214
MOPV036 – Porting Control System Software From Python 2 to 3 - Challenges and Lessons . . . 217
MOPV037 – ALBA Controls System Software Stack Upgrade . . . 222
MOPV039 – UCAP: A Framework for Accelerator Controls Data Processing @ CERN . . . 230
MOPV040 – Introducing Python as a Supported Language for Accelerator Controls at CERN . . . 236
MOPV041 – Modernisation of the Toolchain and Continuous Integration of Front-End Computer Software at CERN . . . 242
MOPV042 – PLCverif: Status of a Formal Verification Tool for Programmable Logic Controller . . . 248
MOPV043 – CERN Controls Configuration Service - Event-Based Processing of Controls Changes . . . 253
MOPV044 – Lessons Learned Moving from Pharlap to Linux RT . . . 257
MOPV045 – Data-Centric Web Infrastructure for CERN Radiation and Environmental Protection Monitoring 261
MOPV047 – Upgrading Oracle APEX Applications at the National Ignition Facility . . . 267
MOPV048 – Fast Multipole Method (FMM)-Based Particle Accelerator Simulations in the Context of Tune Depression Studies . . . 271
MOPV049 – Standardizing a Python Development Environment for Large Controls Systems . . . 277
MOPV050 – DevOps and CI/CD for WinCC Open Architecture Applications and Frameworks . . . 281
TUAL02 – Development of a Single Cavity Regulation Based on microTCA.4 for SAPS-TP . . . 286
TUAL03 – R&D Studies for the Atlas Tile Calorimeter Daughterboard . . . 290
TUAR01 – Upgrade of the CMS ECAL Detector Control System During the CERN Large Hadron Collider Long Shutdown II . . . 297
TUAR03 – The Control System of the Linac-200 Electron Accelerator at JINR . . . 302
TUBL01 – Distributed Caching at Cloud Scale with Apache Ignite for the C2MON Framework . . . 307
TUBL02 – Implementing an Event Tracing Solution with Consistently Formatted Logs for the SKA Telescope Control System . . . 311
TUBL03 – Tango Controls RFCs . . . 317
TUBL04 – CI-CD Practices at SKA . . . 322
TUBL05 – Pysmlib: A Python Finite State Machine Library for EPICS . . . 330
TUBR01 – Nominal Device Support (NDSv3) as a Software Framework for Measurement Systems in Diagnostics . . . 337
TUBR02 – Design Patterns for the SKA Control System . . . 343
TUBR03 – Control System for 6 MeV Linear Accelerator at LINAC Project PINSTECH . . . 348
TUBR04 – Control System of Cryomodule Test Facilities for SHINE* . . . 353
TUPV001 – The Mirror Systems Benches Kinematics Development for Sirius/LNLS . . . 358
TUPV002 – Motion Control Improvements for the Kirkpatrick-Baez Mirror System for Sirius/LNLS EMA Beamline . . . 362
TUPV003 – The Control System of the Four-Bounce Crystal Monochromators for SIRIUS/LNLS Beamlines 365
TUPV004 – The FPGA-Based Control Architecture, EPICS Interface and Advanced Operational Modes of the High-Dynamic Double-Crystal Monochromator for Sirius/LNLS . . . 370
TUPV005 – OPC-UA Data Acquisition for the C2MON Framework . . . 376
TUPV006 – Control System of the SPIRAL2 Superconducting Linac Cryogenic System . . . 382
TUPV007 – Motorized Regulation Systems for the SARAF Project . . . 387
TUPV009 – OpenCMW - A Modular Open Common Middle-Ware Library for Equipment- and Beam-Based Control Systems at FAIR . . . 392
TUPV010 – Integration of OPC UA at ELBE . . . 400
TUPV011 – Interfacing EPICS and LabVIEW Using OPC UA for Slow Control Systems . . . 405
TUPV012 – Automated Device Error Handling in Control Applications . . . 408
TUPV013 – Back End Event Builder Software Design for INO Mini-ICAL System . . . 413
TUPV014 – Control System of a Portable Pumping Station for Ultra-High Vacuum . . . 418
TUPV015 – EPICS Based High-Level Control System for ESS-ERIC Emittance Measurement Unit Device 423
TUPV016 – Design and Development of the New Diagnostics Control System for the SPES Project at INFN-LNL . . . 428
TUPV019 – Control System for 30 keV Electron Gun Test Facility . . . 433
TUPV020 – Automatic RF and Electron Gun Filament Conditioning Systems for 6 MeV LINAC . . . 437
TUPV025 – Control System of Upgraded High Voltage for Atlas Tile Calorimeter . . . 443
TUPV027 – EPICS DAQ System for Beam Position Monitor at the KOMAC Linac and Beamlines . . . 447
TUPV028 – The Control and Archiving System for the Gamma Beam Profile Station at ELI-NP . . . 450

TUPV030 – Redesign of the VELO Thermal Control System Forfuture Detector Development 454
TUPV031 – LHC Vacuum Supervisory Application for Run 3 459
TUPV032 – Challenges of Automating the Photocathode Fabrication Process at CERN 464
TUPV033 – Distributed Transactions in CERN's Accelerator Control System 468
TUPV034 – Development of an Automated High Temperature Superconductor Coil Winding Machine at CERN 473
TUPV035 – Continuous Integration for PLC-based Control System Development 478
TUPV036 – An Evaluation of Schneider M580 HSBY PLC Redundancy in the R744 System A Cooling Unit 484
TUPV037 – Modular Software Architecture for the New CERN Injector Wire-Scanners 487
TUPV039 – A Reliable Monitoring and Control System for Vacuum Surface Treatments 492
TUPV040 – A Python Package For Generating Motor Homing Routines 497
TUPV042 – Collision Avoidance Systems in Synchrotron SOLEIL 501
TUPV046 – Modification of Data Acquisition System in HLS-II Experimental Station 506
TUPV047 – Controlling the CERN Experimental Area Beams 509
TUPV048 – Updates and Remote Challenges for IBEX, Beamline Control at ISIS Pulsed Neutron and Muon Source 514
TUPV049 – The IBEX Script Generator 519
TUPV050 – Control System Upgrade of the High-Pressure Cell for Pressure-Jump X-Ray Diffraction . . . 524
WEAL01 – Image Processing Alignment Algorithms for the Optical Thomson Scattering Laser at the National Ignition Facility 528
WEAL02 – A Framework for High Level Machine Automation Based on Behavior Tree 534
WEAL03 – The Status of Fast Obit Feedback System of HEPS 540
WEAR01 – The Tango Controls Collaboration Status in 2021 544
WEAR02 – Adaptations to COVID-19: How Working Remotely Has Made Teams Work Efficiently Together 550
WEAR03 – Agility in Managing Experiment Control Software Systems 553
WEBL01 – FAIRmat - a Consortium of the German Research-Data Infrastructure (NFDI) 558
WEBL02 – Prototype of Image Acquisition and Storage System for SHINE 564
WEBL04 – Manage the Physics Settings on the Modern Accelerator 569
WEBL05 – FAIR Meets EMIL: Principles in Practice 574
WEBR01 – RomLibEmu: Network Interface Stress Tests for the CERN Radiation Monitoring Electronics (CROME) 581
WEBR02 – Towards the Optimization of the Safety Life-Cycle for Safety Instrumented Systems 586
WEBR03 – The Fast Protection System for CSNS Accelerator 593
WEBR04 – Safeguarding Large Particle Accelerator Research Facility- A Multilayer Distributed Control Architecture 596
WEBR05 – Integrated Supervision for Conventional and Machine-Protection Configuration Parameters at ITER 602
WEPV001 – Temperature Control for Beamline Precision Systems of Sirius/LNLS 607
WEPV002 – Position Scanning Solutions at the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS 613
WEPV003 – The Dynamic Modeling and the Control Architecture of the New High-Dynamic Double-Crystal Monochromator (HD-DCM-Lite) for Sirius/LNLS 619
WEPV005 – Experiment Automation Using EPICS 625
WEPV006 – Automated Operation of ITER Using Behavior Tree Semantics 628
WEPV007 – Machine Learning Projects at the 1.5-GeV Synchroton Light Source DELTA 631
WEPV008 – Online Automatic Optimization of the Elettra Synchrotron 636
WEPV010 – R&D of the KEK Linac Accelerator Tuning Using Machine Learning 640
WEPV011 – Research on Correction of Beam Beta Function of HLS-II Storage Ring Based on Deep Learning 645
WEPV012 – Beam Fast Recovery Study and Application for CAFe 648
WEPV013 – Design of Magnet Measurement System Based on Multi-Hall Sensor 653
WEPV015 – Development of the RF Phase Scan Application for the Beam Energy Measurement at KOMAC 656
WEPV016 – The Automatic LHC Collimator Beam-Based Alignment Software Package 659
WEPV018 – The Linac4 Source Autopilot 665
WEPV019 – Renovation of the Beam-Based Feedback Controller in the LHC 671
WEPV020 – Learning to Lase: Machine Learning Prediction of FEL Beam Properties 677
WEPV021 – Machine Learning for RF Breakdown Detection at CLARA 681

WEPV022 – Sample Alignment in Neutron Scattering Experiments Using Deep Neural Network 686
WEPV023 – Development of a Smart Alarm System for the CEBAF Injector 691
WEPV024 – X-Ray Beamline Control with Machine Learning and an Online Model 695
WEPV025 – Initial Studies of Cavity Fault Prediction at Jefferson Laboratory 700
WEPV026 – Multi-Channel Heaters Driver for Sirius Beamlines Optical Devices 705
WEPV027 – Expandable and Modular Monitoring and Actuation System for Engineering Cabinets at Sirius Light Source 710
WEPV028 – CompactRIO Custom Module Design for the Beamline's Control System at Sirius 715
WEPV031 – Status of the uTCA Digital LLRF design for SARAF Phase II 720
WEPV033 – Architecture of a Multi-Channel Data Streaming Device with an FPGA as a Coprocessor . . 724
WEPV034 – Equipment and Personal Protection Systems for the Sirius Beamlines 729
WEPV036 – The LMJ Target Chamber Diagnostic Module 734
WEPV037 – Development of a Voltage Interlock System for Normal-Conducting Magnets in the Neutrino Experimental Facility at J-PARC 738
WEPV038 – Performance Verification of New Machine Protection System Prototype for RIKEN RI Beam Factory 742
WEPV039 – Novel Personnel Safety System for HLS-II 746
WEPV040 – Design of Machine Protection System for SXFEL-UF 750
WEPV041 – Implementation of a VHDL Application for Interfacing Anybus CompactCom 755
WEPV042 – Applying Model Checking to Highly-Configurable Safety Critical Software: The SPS-PPS PLC Program 759
WEPV044 – Beam Profile Measurements as Part of the Safe and Efficient Operation of the New SPS Beam Dump System 764
WEPV047 – Supporting Flexible Runtime Control and Storage Ring Operation with the FAIR Settings Management System 768
WEPV048 – An Archiver Appliance Performance and Resources Consumption Study 774
WEPV049 – Controls Data Archiving at the ISIS Neutron and Muon Source for In-Depth Analysis and ML Applications 780
THAL01 – Machine Learning Tools Improve BESSY II Operation 784
THAL02 – Bayesian Techniques for Accelerator Characterization and Control 791
THAL03 – Machine Learning Based Middle-Layer for Autonomous Accelerator Operation and Control . . 797
THAL04 – Machine Learning Based Tuning and Diagnostics for the ATR Line at BNL 803
THAR01 – MINT, an ITER Tool for Interactive Visualization of Data 809
THAR03 – Automated Scheduler Software Based on Metro UI Design for MACE Telescope 814
THBL01 – Control System Management and Deployment at MAX IV 819
THBL02 – Exploring Alternatives and Designing the Next Generation of Real-Time Control System for Astronomical Observatories 824
THBL03 – The State of Containerization in CERN's Accelerator Controls 829
THBL04 – Kubernetes for EPICS IOCs 835
THBR01 – Renovation of the Trigger Distribution in CERN's Open Analogue Signal Information System Using White Rabbit 839
THBR02 – White Rabbit and MTCA.4 use in the LLRF upgrade for CERN's SPS 847
THBR03 – Prototype of White Rabbit Based Beam-Synchronous Timing Systems for SHINE 853
THPV001 – Supervisory System for the Sirius Scientific Facilities 858
THPV004 – Open-Hardware Knob System for Acceleration Control Operations 861
THPV005 – Virtual Reality and Control Systems: How a 3D System Looks Like 864
THPV006 – Design of Real-Time Alarm System for CAFe 867
THPV007 – Fast Creation of Control and Monitor Graphical User Interface for PEPC of Laser Fusion Facility Based on ICSFF 871
THPV009 – Web Gui Development and Integration in Libera Instrumentation 875
THPV010 – Scaling Up the ALBA Cabling Database and Plans to Turn into an Asset Management System 878
THPV011 – Notifications with Native Mobile Application 883
THPV012 – LHC Collimation Controls System for Run III Operation 888
THPV013 – WRAP - A Web-Based Rapid Application Development Framework for CERN's Controls Infrastructure 894
THPV014 – Adopting PyQt for Beam Instrumentation GUI Development at CERN 899

THPV015 – New Timing Sequencer Application in Python with Qt - Development Workflow and Lessons Learnt . . . 904
THPV021 – TATU: A Flexible FPGA-Based Trigger and Timer Unit Created on CompactRIO for the First Sirius Beamlines . . . 908
THPV022 – MRF Timing System Design at SARAF . . . 912
THPV025 – A New Timing System for PETRA IV . . . 916
THPV027 – Application of the White Rabbit System at SuperKEKB . . . 919
THPV028 – Analysis of AC Line Fluctuation for Timing System at KEK . . . 923
THPV029 – Development of Timing Read-Back System for Stable Operation of J-PARC . . . 927
THPV031 – Upgrade of Timing System at HZDR ELBE Facility . . . 931
THPV032 – The Demonstrator of the HL-LHC ATLAS Tile Calorimeter . . . 935
THPV033 – Reusable Real-Time Software Components for the SPS Low Level RF Control System . . . 939
THPV036 – Laser Driver State Estimation Oriented Data Governance . . . 942
THPV037 – The Implementation of the Beam Profile Application for KOMAC Beam Emittance . . . 947
THPV038 – Plug-in-Based Ptychography & CDI Reconstruction User Interface Development . . . 950
THPV040 – New Machine Learning Model Application for the Automatic LHC Collimator Beam-Based Alignment . . . 953
THPV041 – Innovative Methodology Dedicated to the CERN LHC Cryogenic Valves Based on Modern Algorithm for Fault Detection and Predictive Diagnostics . . . 959
THPV042 – Evolution of the CERN Beam Instrumentation Offline Analysis Framework (OAF) . . . 965
THPV043 – Using AI for Management of Field Emission in SRF Linacs . . . 970
THPV046 – Virtualized Control System Infrastructure at LINAC Project, PINSTECH . . . 975
THPV047 – Status of High Level Application Development for HEPS . . . 978
THPV048 – Novel Control System for the LHCb Scintillating Fibre Tracker Detector Infrastructure . . . 981
THPV049 – Virtualisation and Software Appliances as Means for Deployment of SCADA in Isolated Systems 985
FRAL01 – The Laser MegaJoule Facility Status Report . . . 989
FRAL02 – DISCOS Updates . . . 994
FRAL03 – CERN Cryogenic Controls Today and Tomorrow . . . 997
FRAL04 – The Control System of the New Small Wheel Electronics for the Atlas Experiment . . . 1005
FRAL05 – MACE Camera Electronics: Control, Monitoring & Safety Mechanisms . . . 1011
FRAR01 – Taranta, the No-Code Web Dashboard in Production . . . 1017
FRAR02 – canone3: A New Service and Development Framework for the Web and Platform Independent Applications* . . . 1023
FRAR03 – A Major Update of Web Based Development Toolkit for Control System of Large-Scale Physics Experiment Device . . . 1029
FRBL01 – Machine Learning for Anomaly Detection in Continuous Signals . . . 1032
FRBL03 – A Literature Review on the Efforts Made for Employing Machine Learning in Synchrotrons . . . 1039
FRBL05 – RemoteVis: An Efficient Library for Remote Visualization of Large Volumes Using NVIDIA Index 1047
FRBR01 – Process Automation at SOLEIL: Two Applications Using Robot Manipulators . . . 1054
FRBR02 – An Integrated Data Processing and Management Platform for X-Ray Light Source Operations* 1059
FRBR03 – Status of Bluesky Deployment at BESSY II . . . 1064
FRBR04 – Continuous Scans with Position Based Hardware Triggers . . . 1069

Appendices **1075**
List of Authors . . . 1075
Institutes List . . . 1087

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

MATURITY OF THE MAX IV LABORATORY IN OPERATION AND PHASE II DEVELOPMENT

V. Hardion, P. Bell, M. Eguiraun, T. Eriksson, A. Freitas,
M. Klingberg, M. Lindberg, Z. Matej, S. Padmanabhan, A. Salnikov,
P. Sjöblom, D. Spruce, MAX IV Laboratory, Lund University, Lund, Sweden

Abstract

MAX IV Laboratory, the first 4th generation synchrotron located in the south of Sweden, entered operation in 2017 with the first three experimental stations. In the past two years the project organisation has been focused on phase II of the MAX IV Laboratory development, aiming to raise the number of beamlines in operation to 16. The KITS group, responsible for the control and computing systems of the entire laboratory, was a major actor in the realisation of this phase as well as in the continuous up-keep of the user operation. The challenge consisted principally of establishing a clear project management plan for the support groups, including KITS, to handle this high load in an efficient and focused way, meanwhile gaining the experience of operating a 4th generation light source. The momentum gained was impacted by the last extensive shutdown due to the pandemic and shifted toward the remote user experiment, taking advantage of web technologies. This article focuses on how KITS has handled this growing phase in term of technology and organisation, to finally describe the new perspective for the MAX IV Laboratory, which will face a bright future.

MAX IV GENERAL STATUS

MAX IV Laboratory [1] is a synchrotron based research facility which consists of two storage rings of 1.5 GeV and 3 GeV respectively fed by a full energy linear accelerator. These two rings provide X-rays to 16 beamlines, of which 14 are today in user operation (see Fig. 1) and the remaining two will come online during the period 2022-2023.

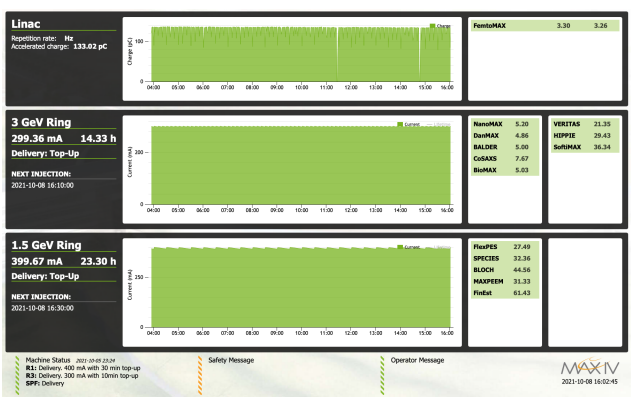


Figure 1: The Machine Status web application accessible for the public, showing all accelerators and beamlines in operation.

Beamlines Status

The first phase beamlines have rolled out into their normal operation and the number of articles published by MAX IV compared to the previous MAX-Lab are rising (Fig. 2). Beamlines already in operation have increased their performance above the baseline, aided by the support from the KITS group.

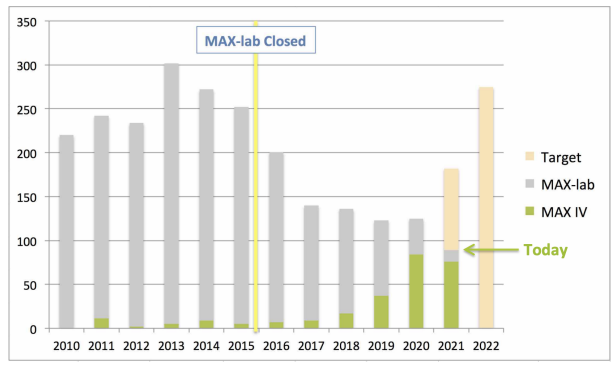


Figure 2: Publication of MAX IV Laboratory vs MaxLab.

For example Balder has achieved a unique performance with a continuous scanning down to 30 s per EXAFS in combination with a powerful analysis software [2], opening up more experiments for environment science.

In the meantime the beamlines which started in 2020 are getting excellent commissioning results while welcoming expert user experiments, in preparation for general users. Four new beamlines have completed the portfolio of full user operation since then. After achieving baseline requirements, FemtoMAX could open its first user call with a time precision of 250 fs and enough signal to complete an experiment in a standard beamtime. Thanks to large resource investment, the X-Ray pulse of the LINAC has been upgraded to 10 Hz operation while on the beamline, a precise data acquisition system has been developed by KITS to acquire all the precious shots while applying a time-over-threshold computation within 10 ms.

In 2021 the new beamlines have started operation early 2021 with limited offer. These new phase II beamlines are increasing the global level of efficiency by profiting on new standards using more stable equipment such as the PandABox [3]. DanMAX has just finished the commissioning phase and started to accept expert Users for the PRXD experiment. In order to guarantee the expected performance of the experiment, KITS have developed a position-based hardware triggered continuous scan [4]. COSAXS started early in 2021 with a basic SAXS experiment based on time

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

constant continuous scan with a Eiger 4M detector. The unique Coherence properties of a 4th generation synchrotron light has been demonstrated [5] on CoSAXS which will also open unique XPCS experiments in few years. In continuous 6 months pace development, the beginning of the year brought time resolved capabilities, while WAXS is expected to be available by the end of the year.

Accelerator Status

The 3 accelerators have been in operation since 2017 and they are continuously improving their characteristics with examples being the stability in the storage rings and the increase of repetition rate in the linear accelerator from 2 Hz to 10 Hz.

Several projects have been carried out over the past years such as X-ray beam position monitors on the front end of beamlines. New corrector magnets have been deployed and feed-forward control loops implemented to compensate the distortion in the beam orbit due to the motion of the undulators. To characterize the longitudinal phase space and slice parameters for the linac, a Transverse Deflecting Cavity (TDC) system has been developed and partly installed. This new system gives valuable information to tune beam parameters. These are some of many other projects to improve the stability and beam quality of the MAX IV accelerator.

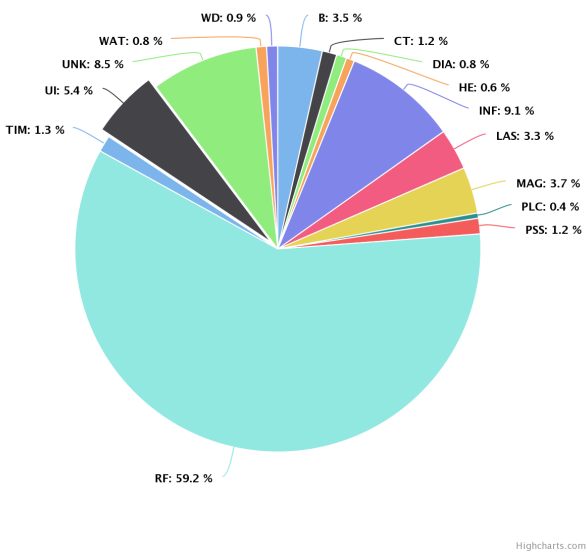


Figure 3: The downtime of MAX IV from 2021-01-01 till 2021-10-06. Biggest downtime corresponds to RF system due to the long recovery time for RF cavity conditioning.

Since October 2020 combined Slow Orbit Feedback (SOFB) and Fast Orbit Feedback (FOFB) has been used during delivery in the 3 GeV ring, a key milestone in the machine for reaching its design goals in terms of beam stability. This system was developed between KITS and the Accelerator Development group and was the final step in this project following the implementation of a purely Tango [6] software based SOFB, running at 10 Hz on the VM infrastructure [7]. The FOFB is implemented in custom hardware with a 10 kHz feedback loop but communicates via Tango with the SOFB in order to periodically offload the cumulative effects of the fast correctors to avoid their saturation.

The operators involved in the daily user operation have tracked and classified all the beam interruption via a web application [8]. Based on the accumulated data, and since 2020, the main efforts have been focused on tracking the Mean Time Between Failure (MTBF) which has led to different actions on the Machine and Beamline Protection System, specially to avoid human error. In 2021, the major downtime which requires a longer Mean Time To Repair corresponded to the RF section of the LINAC, mostly due to the inherent nature of time consuming conditioning, see Fig. 3. An application for accelerator failure prevention based on Machine Learning was develop to track the non obvious causes [9].

The total delivery plan for all MAX IV accelerators from the period 2021-01-01 till 2021-10-06 was 7104 hours, with a downtime of 310.68 hours. The control system represented 1.2 % of this downtime (3.72 hours) due to a control system failure over all three accelerators.

KITS Improvement

In terms of development an important effort has been made in the last two years in a quite large number of projects. Partly due to the increased capabilities due to the growing numbers of staff, but also due to the demanding request coming from our stakeholders and the necessity of gaining knowledge to better support MAX IV needs. The following activities are the most remarkable ones.

Hardware Standard MAX IV has entered in the PandABox collaboration started by Soleil and Diamond synchrotron facilities. This has become a standard with so far, 20 units manufactured and 15 units deployed in the facility mainly at the endstations. The main purpose of the PandABox is synchronized triggering as they can receive time stamped bunch clock triggering from accelerators and fan it out with configurable delay. The PandABox are also often equipped with a D-tAcq 1 MSps 18 bit analog input and 25 018 kSpsbit 16 bit analog output card integrated into the 1u 19 inch rack.

With maintaining the philosophy that electronics are closely integrated with the control system software, which is a collaborative environment at MAX IV, much of the work revolves around collaborative projects, e.g. the IcePAP motor driver [10] and Em# electrometer [11].

Monitoring On the Hardware part, important and labor-intensive task moving into operation of a synchrotron facility is continuously monitoring and refining performance of commissioned equipment. This is done in a systematic way as e.g. described in [12] for monochromators at the soft x-ray beamlines and in [13] for many of the mirror chambers in the facility.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

In general a substantial effort has been focused on the monitoring infrastructure using Prometheus and the Elastic Stack (Filebeat, Logstash, Graphana). The alarm system has the potential to profit from this ecosystem and some projects like the notification system by Android or IOS [14].

Deployment In terms of software infrastructure Conda [15] is also used for the deployment when system dependency does not fulfill the requirement of the application. As no decision has been made on the CentOS 8 replacement after the announce of Red Hat to change its scope, the current CentOS 7 became quickly obsolete. In addition, the continuous integration has been fully migrated from Jenkins to Gitlab pipeline.

Detector and Streaming Particle Detectors are the most complex piece of hardware to integrate in the experimental stations. Previously a reliance on reusing existing software meant that diagnosing problems in operation was difficult due to the lack of knowledge of the detector systems. In 2019, 2 Full Time Equivalents (FTE) were dedicated to increase the group knowledge in Particle Detectors. The major development was to focus on the data retrieval in order to match the performance of the detector and slowly gain expertise in this challenging domain.

Approximately half the KITS software effort is now spent on data acquisition (DAQ) mainly on high frame and data rate photon counting detectors. There is an ongoing effort to unify all detectors under a common data and metadata streaming standard. In term of infrastructure the architecture has been shifted from file exchange paradigm to streaming data with ZMQ over 40 Gb/s links to a central DAQ cluster (see next section). The data can be consumed directly by online processing pipelines which simplify the all process. In combination with this activity, the KITS group has validating the physics performance of the detectors, for example in site acceptance tests, and offering the beamlines support in case of issues, for example in liaising with the companies.

Continuous Scanning The stability of the scanning system has increased by mastering the behaviour of the detectors and simplification of the data retrieval. In order to deliver the top performance, the continuous scanning of the sample has become the standard implementation and the development has been streamlined by the use of the PandABox to manage the synchronisation between motion, triggers and I/O, consequently making the previously used delay generators and counter cards obsolete.

The accelerators have continuously made a series of changes to the insertion devices described in [16] to allow for synchronized motion between monochromators and insertion devices. With this feature fully implemented the insertion device will be integrated into the overall scan capability.

Taranta In the User Interface (UI) domain, web technology has started to replace the traditional desktop applications. Inspired by Taurus, a Taranta application [17] allows the user to compose themselves their dashboards by drag and drop of basic widgets. The resulting UI is immediately usable in the real control system.

Remote Operation During the pandemic the laboratory has been in limited operation. Remote users were supported by beamline staff working extra hours for tasks that are not automated, for example mounting samples. In some limited examples like Macromolecular Crystallography (MX), the MXCuBE web application is already prepared and facilitates the users remote interaction with the beamline. Simultaneously, a large effort went into making safe use of the hardware in remote operation on other beamlines similarly to the BioMAX beamline, which was MAX IVs first beamline to run fully remotely. For other cases, a low cost solution via remote desktop has been used for bringing the experiment control room to the users home, and being able to share the experience with a user elsewhere. Additional steps have been made to ensure the safety of the remote operation, i.e. prohibiting certain motions from happening and, preventing users from accessing critical control system parameters while being remote.

Kubernetes Clusters

Various systems at MAX IV are relying on containerization technologies for a flexible and reproducible way of application management. This includes classical web-applications, computing workloads and data acquisition pipelines. With a growing demand on running containerised workloads, introducing the container orchestration was a step forward for the facility. Currently the IT infrastructure team at MAX IV is operating several on-premises Kubernetes (K8S) clusters, targeting different application vectors:

- General Services cluster based on OKD, the community upstream of RedHat OpenShift, is targeting web-services (e.g. wiki, AWX, Taranta, etc.).
- DAQ cluster with high performance bare-metal worker nodes is used for acquiring the experimental data (Fig. 4). The nodes are connected to a high-speed 40 Gbps network fabric for data ingest, another 40 Gbps for streaming data out and to the IBM Spectrum Scale (GPFS) filesystem via InfiniBand fabric.
- JupyterHub cluster nodes with GPU accelerators are used for high performance processing. Similar architecture than the DAQ cluster specific application-level feature (LXCFS) improves the user experience of resource visibility.

Infrastructure side also includes two shared HAProxy load-balancer machines with Keepalived to route the HTTP traffic mainly towards the K8S Ingress controllers.

MAX IV chose Helm Chart and GitLab CI pipelines for the application deployments on K8S. For example the “daq-pipeline” Helm Chart creates the necessary set of resources to start the data acquisition for the specified beamline, the detector and DCU network endpoints.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

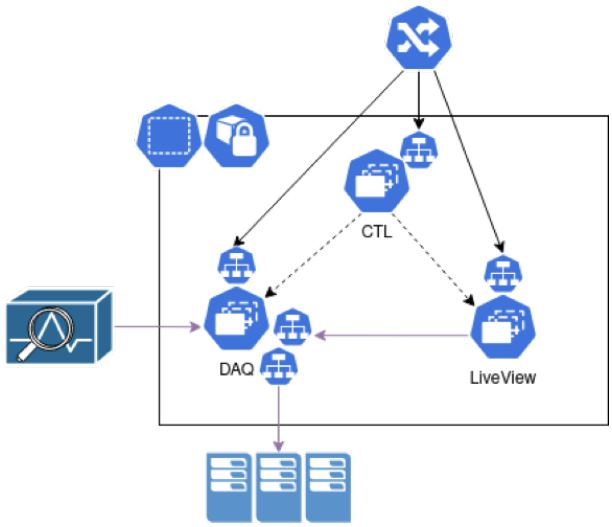


Figure 4: The kubernetes architecture for the Data Acquisition Cluster.

Scientific Software

With the improvement brought by the 4th generation synchrotron, many experiments have improve the rate of their data acquisition moving the bottleneck to preparatory phase or data processing. In Macromolecular X-ray crystallography (MX) an automatic sample centring would streamline the process avoiding human intervention which is possible with a Machine Learning processing [18]. Another development in MX, FragMax [19] has been developed to speed up the interpretation of the results and to facilitate processing by running multiple user configured analysis on a set of datasets. In Small Angle X-ray Scattering and Power Diffraction, the Azimuthal Integration is the first qualitative appreciation of the raw data. A real time processing with FPGA would give substantial advantage for the User [20].

Data Management

SciCat is database together with a web user interface to display information for the data acquisitions performed by a user. It includes a detailed set of metadata information for adding appropriate context to the experiments. Metadata keys range from common beamline parameters, the particular experiment setup but also free comments added by the user. It is under development as part of a collaboration with ESS and PSI facilities. A companion application has been developed in MAX IV, Scanlog, that provides a different user interface on top of exisiting Scicat database, the main different is that this new ui is desgined to provide a quick overview of the scans during the beamtime. It also provides some filtering so the user can customize the view. It is also possible to rate your data and write comments for a quick overview of if a scan was successful or not.

SciDog is used to choose which metadata should be stored. The beamline staff in this case, can save a configuration with selected metadata, or use all the device/attribute combinations that are in the database. This metadata is gathered mostly by specific Sardana recorders, but there is also ongoing development to allow other beamlines to store metadata that do not rely on Sardana for data acquisition, e.g. commercial spectroscopy solutions.

KITS OPERATION

Having three accelerators and 14 beamlines in operation has been a major change for the KITS group, together with the shift of focus toward a stable and reliable system.

The Controls groups are responsible for the development and maintenance of the experimental system formed by the stack from hardware to software component. For the operation, a system based on Kanban handle a continuous flow of high priority support tasks, representing 15 % of the Controls groups resources. While other operational maintenance represented 25 % of the resource, the figure has even rocketed during the pandemic period to rose above 50 %.

A support system has been established to answer rapidly by phone to any blocking issue regarding user operation. Each call is registered in the electronic logbook Elogy [21] in order to follow up on any critical issues.

In parallel a local contact system follows up closely the issues that arose during beam delivery. When the resolution is minor, the contact can fix it directly by documenting (case UI), by the reconfiguration the system, or by diagnosing a more complex issue. The role of this contact is important to understand the context and usage of the Control system at each beamlines and accelerators. Issues are raised to a two weeks plan revision when it involves more than one resource groups or days of work.

Main Issues

The lab-wise issues are not quantified in regards of User beam time loss, nevertheless the issues are mainly due to the maturity of the system. The more a system is used in production, the more corner cases are solved. But the challenge comes when many beamline are operating while introducing changes e.g. a new usage, a new firmware, a new model of equipment, etc. which is often a necessity for the development of the new features. Others cause have been identified.

Priority on the Feature by the product owner or,in other cases, quality requirement are skipped by the accumulated delay in the project with fixed deadline. Since the development in the project phase is mainly focusing on new features, the new system appears in operation with its most optimised "happy" path.

Behaviour of the Equipment is the cause of many instabilities, mainly visible through the data acquisition system. After the strong focus on detectors, the focus has shifted in 2021 toward the scanning orchestration system which is the main interface for the user to generate experimental data.

User Friendliness is one of the main weakness of the current software solution which generates confusion by hiding the true cause of failure. Thus the importance of the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

software group and the KITS contact to be present during the commissioning phase and the first experts users experiments, until the usage forces the non-nominal cases to surface. Additionally Software developers, by knowing the intrinsic details of the system, can easily accommodate with a complex user interface [22].

Resource of the SW group was impacted by a huge turnover while the projects management estimated 160 % of SW resource to reach the new beamlines in operation. A successful recruitment campaign was established to increase the SW staff from 12 to 18 while maintaining the effort with 4 FTE from the consultant companies S2Innovation and DVel.

KITs Operations Service (KITOS)

From summer 2021 the operation support is increasing its quality of service in order to minimise the downtime of User operation. Due to the restriction of time many project were focus on the functionality of the system rather than the non-functional side like the usability and the reliability of the system, which is usual focus for the commissioning period. By consequence the operation, especially on the new beamlines, suffers from the all the non nominal cases which need a reactive support. The phenomena was multiplied by the new beamlines coming in operation at the same time. The follow up of the main important cases could help to improve the situation although the effort to cover all the cases often required synchronisation of several stakeholders. Previous choice of standard are being reconsidered to obtain a sustainable maintenance, while other are improve to follow the expectation above baseline.

On the critical path of generating data the scanning and data acquisition system is highly sensitive to the weakest element in the chain. Since most of the time the 3rd party components are hardly possible to improve, a large effort is being roll-out to increase the robustness of the Sardana, appearing like the top of the iceberg.

The concept for a KITs Operations Service (KITOS) will provide a stronger focus and wider access to expertise through a single number on support from the KITS group. Inspired by the EuXFEL this service will be dedicated only to supporting actual user operation which includes accelerator operational support, beam line commissioning and setup for experiments, but not the daily work which consists of tasks and project development activities.

The support crew of the week will be covered by two shift crews consisting of 2 people each. Members of the shift crew will be drawn primarily from the KITS team and will consist of a shift leader and shift deputy. During the initial phase it will be important to look for complementary pairings in expertise and experience for the crew, for example experienced and inexperienced. Ideally the shift crew will perform shifts over the on call period to provide continuity.

A KITS Run Coordinator (KRC) will be assigned over one week periods. The role of the KRC will be to support the KITOS and facilitate communication on a wider basis which will allow the KITOS to prioritise supporting the accelerator and beamlines for the running experiments. Issues which arise that require more time and involvement from several parties, possibly with the scheduling of ad-hoc meetings will be addressed by the KRC. In general, the KRC cannot be a member of the shift crew while they are acting as KRC, although in cases where a shift crew member is unable to perform their duties, the KRC may step in until an alternative solution is found. KRC will attend important meetings for reporting.

The shift leader will be responsible for operational decision-making related to KITS systems during x-ray delivery, but shall seek advice from 2nd level oncall in cases of doubt. If the shift leader does not feel able to take a certain operational decision, this decision may be escalated to the KITS run coordinator. The next, and final level of escalation will be the head of KITS, or appointed substitute.

PROJECT MANAGEMENT

The new Central Project Office (CPO) project management organisation made visible the resource demanding schedule for the beamlines' projects . With the new time plan established in 2019 the projects responsible have been able to identified what and when the resource groups could realistically work (Fig. 5). After the identification of several bottlenecks, a lab-wise priority was determined to avoid competing resource and unsustainable deadlines.

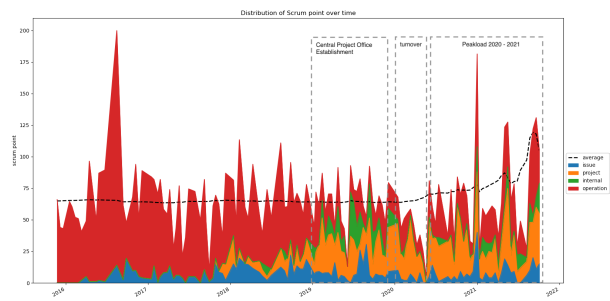


Figure 5: Evolution of the workload distribution over the years. In 2019 the new project management was established.

The new approach breaks down the projects into limited scope i.e optic installation, 1st experimental set up, etc., making the request of resource more formal. The control of feature and time at the beginning of the project introduces an overhead which become less insignificant along the timeline of the project. The enumerated specifications help to define the close-out of the project, freeing resource for the next one. On the overall the gain allows to efficiently manage the resource groups.

On the other hand small jobs don't fit the project schema, becoming very bureaucratic. Also simple request can trigger unnecessary long discussion and often complex organisation of the conception, instead of reusing the standard solution with known but acceptable drawbacks. For this purpose these small requests are routed toward the operation management with much less overhead. Newly created in 2020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the Beamline Project Advisory Group (BPAG), a group of Science Division representatives, acts as toll gate to manage the priority and appoint a dedicated manager to follow up the request with the resource groups. It's expected to streamline the process by adapting step by step the process of each resource group to the operation management.

KITS Integration

The KITS Controls groups were involved in 28 priority projects in 2020 and 21 in 2021. The projects portfolio has included two beamlines delivered for operation per year representing approximately 25 % of the work load while the improvement of the beamlines in operation and the accelerators above baseline representing respectively 50 % and 25 %. The tight schedule to deliver the new beamlines ready for user operation in their baseline configuration was the main concerning challenge.

The new phase II beamlines projects have profited a lot of the standardisation and improvement of theirs optics section from the phase I beamlines. In return the new development has improved the current infrastructure which has indirectly profited to the system already in operation.

From the Control System point of view the new project management organisation settled in 2019 has allowed to increase to 1 year the visibility in term of minimal requirement and priority of projects. The development could foresee a shift in domain from control and monitoring to data acquisition.

A first approximation of the workload, made by comparing the new plan in 2020 helps to identify a 75 % (250 % including low priority request) increase of work load mainly due to the increase of required feature above baseline design of the system.

On the macroscopic view the waterfall based CPO process has been replaced by a more iterative approach in 2020, although still keeping a focus on requirement and specification in order to define the backbone milestone. The KITS Controls activities arrived late in the overall process which the Agile methodology could not compensate the risk of unclear requirement.

A lot of effort has been initiated to make compatible the global management and the local Control System project management. Figure 6 introduces a streamlined process in which a stronger focus is put on requirements and validation. In this sense, a new key role has been introduced, the beamline software owner, who is the main contact for KITS Controls for a given project, and it also must ensure that validations can be properly scheduled and completed, an issue that was not the case in the past in all cases.

In this sense the project managers have been flexible to accommodate their process to the know-unknown. For example the commissioning of the beamline end station has now been integrated into the project process which helps the quality in order to make operable a minimum viable product, although the constraint of the User call deadline at the end of the project does not cover for a full user friendly system.

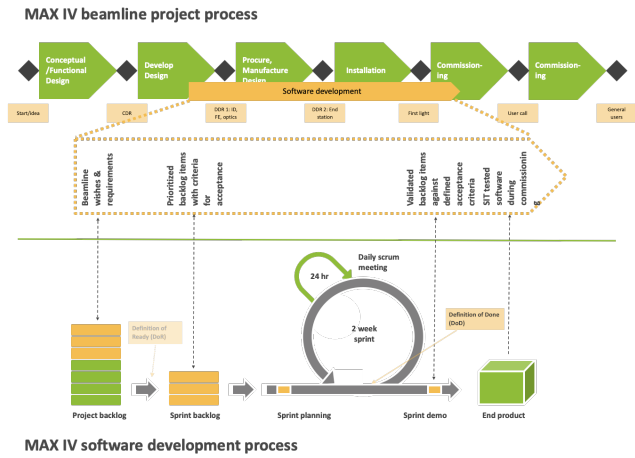


Figure 6: Workflow of SW development integrated in the global management.

During the COVID pandemic the Agile process had accumulated over several months many new features which were awaiting for validation from the requestors mainly due to the low availability of beam time. This has decreased the efficiency of the Agile method based on rapid feedback.

Lessons Learned

The role of CPO to determine where we invest effort was definitely needed, even if it brings an overhead. It is a price to pay but we would have had difficulties without them. Looking at lessons learned regarding the CPO project structure, it is based on a fixed scope and allows very few deviations which fits perfectly with installation projects. On the other hand experimental projects need flexibility for the development of very new features. Change management with impact analysis may improve the process if working on a fixed budget and time. The CPO project management has also improved the testing phase to include a longer commissioning time to verify and validate the operation of new systems (2021).

When BPAG and CPO manage projects with a dedicated project manager (PM), the process becomes more efficient contrary to a mixed position with conflicting responsibilities.

Moving away from pure installation, it is increasingly hard to close the projects due to difficulties pinning down the scope. CPO is trying to move towards a more agile way of working. CPO is also raising the understanding among the stakeholders for the importance of timely validation, and adjusting the ambition depending on how complex a feature turns out to be. There have been examples of arbitrary closure/end point without actual commissioning. In reality, the most successful projects are the ones where the KITS contact (a member with very tight contact with the beamline needs) has played an important role as technical project leader, leaving the big picture for the project manager avoiding to be involved in the details.

The unclear responsibility of which resource group the development has been assigned to was a recurrent impediment although not identified as risk in the project organisation but

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

causing a substantial effort. Two majors cases were identified which have caused delays, waste of resource and conflicts.

In the first case causing 3–4 months delay, resource stakeholders were identified at the start of the project but not the responsibility of the subsystem delivery which led to the reconsideration of an important standard. In order to mitigate any future issue, the PM process has been revised to include a risk assessment session to complete the kick off of a project. Also, in parallel, a technical forum has been established to identify in advance any change in the standard portfolio and assess the impact on all the MAX IV resource groups before it appears in a project timeline.

The second case happened when one resource stakeholder were not clearly identified as being part of the project which led to a competitive situation triggered by the request to multiple channels of communication. By passing the project management, the waste of resource was the direct consequence but minor compare to the indirect consequence on the MAX IV organisation. Whereas 2 resource groups should collaborate on a gray zone (shared responsibility), this led to a confuse direction, protective division and resource competition between engineering and scientific concerns for which a cohesion is a fundamental for big research facility such MAX IV. An investigation for a clear organisation of responsibility is in progress in coherence with the latest recommendation from the Swedish Research Council (VR) review committee [23].

FUTURE

SXL FEL

To allow the study of dynamic phenomena at the atomic and molecular scales a X-Ray free-electron laser (FEL) is necessary in order to provide X-ray beams of sufficient intensity and coherence in short enough pulse duration. MAX IV is designing a Soft X-ray Laser (SXL) beamline [24]. SXL consists of four parts: the already operating MAX IV linear accelerator, a new 40 m undulator, a corresponding beamline and a set of experimental stations. SXL thus capitalizes on the capabilities of the MAX IV linear accelerator. SXL will be placed in a building extension of the Short Pulse Facility (SPF), today housing the FemtoMAX beamline.

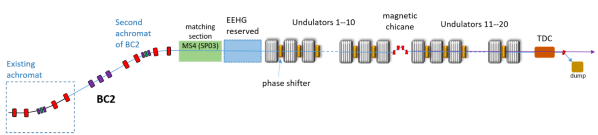


Figure 7: Layout of the SXL FEL line.

The Fig. 7 illustrates FEL SXL with 20 APPLE-X helical undulators, a combined phase shifter and delay chicane. This will trigger updates and new challenges for the MAX IV control system such as a more accurate timing system, synchronized motion control for all undulators, more feedback and feed-forward control loops.

Improve Beamline Capabilities

Among the existing operational beamlines the challenge will be to continue the development of the End Stations' increased capabilities and then new upcoming sample environments. This requires a more streamlined experimental orchestration synchronization with the data acquisition in order to achieve higher rates. Moreover, new sample environments demand a user friendly setup change which in turns requires an increased stability of the overall control system.

Two new beamlines are already in advanced phase of development. They will both bring a high challenge to KITS. MicroMAX [25] aims at improving further data acquisition capabilities for Macromolecular Crystallography (MX) at MAX IV, expanding the current MX capabilities with new SSX techniques as well as a new high performance detector Jungfrau [26]. A high degree of automation an complex web user interfaces are required, as well as increased capabilities of the existing computing infrastructure. ForMAX will offer to the users full-field tomographic imaging with small- and wide-angle x-ray scattering focused on wood related materials. The fast acquisition not only requires the capability of ingesting a high data rate and volume, but is also needs to provide very precise synchronization on the acquisition axis.

New detectors and techniques will require dedicated and specialised efforts, for example:

- measuring and applying count rate correction.
- gaining expertise in FPGA frame grabber for the tomography cameras.
- rethink the DAQ architecture for ever higher rates, e.g. for the Jungfrau with dedicated 100G connection.

Just as trying to standardise the DAQ, all these subjects needs custom solutions and deep knowledge of each system, from the hardware to the scientific processing which challenges the organisation of the KITS group.

ACKNOWLEDGMENT

The authors would like to thank the whole of the Controls and IT (KITS) group, the Beamline Office and the Accelerator Deputy at the MAX IV Laboratory, along with the physicists, engineers and beamline staff, without the collaboration of whom this work would not have been possible. The authors also appreciate the support of the Knut And Alice Wallenberg foundation for funding the DataSTAMP project.

REFERENCES

[1] MAX IV Laboratory home page: http://maxiv.lu.se

[2] J. Murari *et al.*, "Parallel Execution of Sequential Data Analysis", in *Proc. ICALEPCS'17*, Barcelona, Spain, pp. 1877–1879, 2017. doi:10.18429/JACoW-ICALEPCS2017-THPHA186

[3] S. Zhang *et al.*, "PandABox: A Multipurpose Platform for Multi-technique Scanning and Feedback Applications", in *Proc. ICALEPCS'17*, Barcelona, Spain, pp. 143–150, 2017. doi:10.18429/JACoW-ICALEPCS2017-TUAPL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[4] H. Enquist *et al.*, "Continuous Scanning with Position-Based Hardware Triggers at MAX IV beamlines", presented at ICALEPCS'21, Shanghai, China, 2021, paper FRBR04, this conference.

[5] M. Kahnt *et al.*, "Measurement of the coherent beam properties at the CoSAXS beamline", *J. Synchrotron Rad.*, vol. 28, pp. 1948-1953, 2021. doi:10.1107/S1600577521009140

[6] A. Gotz *et al.*, "The TANGO Controls Collaboration in 2015", in *Proc. ICALEPCS2015*, Melbourne, Australia, pp. 585–588, Oct. 2015. doi:10.18429/JACoW-ICALEPCS2015-WEA3O01

[7] P.J. Bell *et al.*, "A General Multiple-Input Multiple-Output Feedback Device in Tango for the MAX IV Accelerators", *Proc. ICALEPCS'19*, New York, NY, USA, pp. 1084–1088, 2019. doi:10.18429/JACoW-ICALEPCS2019-WEPHA012

[8] B. Meirose *et al.*, "Real-time Accelerator Diagnostic Tools for the MAX IV Storage Rings", *Instruments*, vol. 4, no. 3, pages 26, 2020. doi:10.3390/instruments4030026

[9] J. Petersson *et al.*, "Machine learning applications for accelerator failure prevention at MAX IV", presented at ICALEPCS'21, Shanghai, China, 2021, paper THPV039, this conference.

[10] P. Sjöblom *et al.*, "Motion control system of MAX IV Laboratory soft x-ray beamlines", in *Proc. AIP Conference Proceedings*, New York, NY, USA, vol. 1741, p. 030045, 2016. doi:10.1063/1.4952868

[11] J.A. Avila-Abellan *et al.*, "Em# Electrometer Comes to Light", in *Proc. ICALEPCS'17*, Barcelona, Spain, pp. 137–142, 2017. doi:10.18429/JACoW-ICALEPCS2017-TUAPL04

[12] P. Sjöblom *et al.*, "Understanding the mechanical limitations of the performance of soft X-ray monochromators at MAX IV laboratory", in *Journal of Synchrotron Radiation*, vol. 27, no.2, pp. 272–283, 2020. doi:10.1107/S1600577520000843

[13] M. Agåker *et al.*, "A five-axis parallel kinematic mirror unit for soft X-ray beamlines at MAXIV", in *Journal of Synchrotron Radiation*, vol. 27, no. 2, pp. 262–271, 2020. doi:10.1107/S160057751901693X

[14] B. Bertrand *et al.*, "Notifications With Native Mobile Application", presented at ICALEPCS'21, Shanghai, China, 2021, paper THPV011, this conference.

[15] B. Bertrand, *et al.*, "Control System Management and Deployment at MAX IV", presented at ICALEPCS'21, Shanghai, China, 2021, paper THBL01, this conference.

[16] J. Lidon-Simon *et al.*, "Control System Integration of MAX IV Insertion Devices", in *Proc. of ICALEPCS'19*, New York, NY, USA, pp. 525–529, 2019. doi:10.18429/JACoW-ICALEPCS2019-MOPHA132

[17] M. Eguiraun *et al.*, "Taranta, the No-Code Web Dashboard in production", presented at ICALEPCS'21, Shanghai, China, 2021, paper FRAR01, this conference.

[18] I. Lindhé *et al.*, "Automated ML-based Sample Centering for Macromolecular X-Ray Crystallography with MXAimbot", presented at ICALEPCS'21, Shanghai, China, 2021, paper FRBR06, this conference.

[19] G. Lima *et al.*, "FragMAX: the fragment-screening platform at the MAX IV Laboratory", *Acta Cryst.*, vol. D76, pp. 771-777, 2020. doi:10.1107/S205979832000889X

[20] Z. Matej *et al.*, "Real-Time Azimuthal Integration of X-Ray Scattering Data on FPGAs", presented at ICALEPCS'21, Shanghai, China, 2021, paper FRBL04, this conference.

[21] Elogy Electronic Logbook: https://gitlab.com/MaxIV/Elogy

[22] J. Swarts *et al.*, "Open-Source Software in the Sciences: The Challenge of User Support", *Journal of Business and Technical Communication*, vol. 33, pp. 60-90, 2019. doi:10.1177/1050651918780202

[23] 4th Review of MAX IV's Project Management: https://www.maxiv.lu.se/wp-content/plugins/alfresco-plugin/ajax/downloadFile.php?object_id=590715d2-c4c3-4d6f-9660-3d7f87e6eb27

[24] P. Tavares *et al.*, "The Soft X-ray Laser @ MAX IV - Conceptual Design Report", https://www.maxiv.lu.se/soft-x-ray-laser/

[25] A. Shilova *et al.*, "Current status and future opportunities for serial crystallography at MAX IV Laboratory", *J. Synchrotron Rad.*, vol. 27, p. 1095, 2020. doi:10.1107/S1600577520008735

[26] F. Leonarski *et al.*, "JUNGFRAU detector for brighter x-ray sources: solutions for IT and data science challenges in macromolecular crystallography", *Structural Dynamics*, vol. 7, p. 014305, 2020. doi:10.1063/1.5143480

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL02

STATUS OF THE NATIONAL IGNITION FACILITY (NIF) INTEGRATED COMPUTER CONTROL AND INFORMATION SYSTEMS

M. Fedorov, A. Barnes, L. Beaulac, G. Brunton, A. Casey, J. Castro Morales, J. Dixon, C. Estes, M. Flegel, V. Gopalan, S. Heerey, R. Lacuata, V. Miller Kamm, M. Paul, B. Van Wonterghem, S. Weaver, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore 94550, CA

Abstract

The National Ignition Facility (NIF) is the world's most energetic laser system used for Inertial Confinement Fusion (ICF) and High Energy Density Physics (HEDP) experimentation. Each laser shot delivers up to 1.9 MJ of ultraviolet light, driving target temperatures to more than 180 million K and pressures 100 billion times atmospheric, making possible direct study of conditions mimicking interiors of stars and planets, as well as our primary scientific applications: stockpile stewardship and fusion power. NIF control and diagnostic systems allow physicists to precisely manipulate, measure and image this extremely dense and hot matter. A major focus in the past two years has been adding comprehensive new diagnostic instruments to evaluate increasing energy and power of the laser drive. When COVID-19 struck, the controls team leveraged remote access technology to provide efficient operational support without stress of on-site presence. NIF continued to mitigate inevitable technology obsolescence after 20 years since construction. In this talk, we will discuss successes and challenges, including NIF progress towards ignition, achieving record neutron yields in 2021.

INTRODUCTION

The National Ignition Facility is a large (3 football fields) and complex (192 laser beams) experimental physics system (Fig. 1) [1] . It is efficiently operated 24x7 by a shift of the 12-14 Control Room operators with the help of the Integrated Computer Control System (ICCS). Over 66,000 devices with rich APIs are distributed over 2,300 front-end-processors (FEPs) and embedded controllers (ECs).

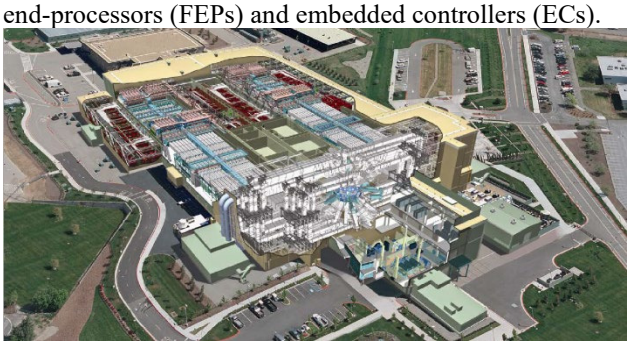


Figure 1: NIF building layout.

NIF experiments are structured around laser shots. Each shot takes 4-8 hours and involves the control system to execute over 2 million device operations.

Experiments at NIF support several programmatic missions, such as Stockpile Stewardship, Discovery Science, National Security Applications, and Inertial Confinement Fusion (ICF). For ICF, thermonuclear ignition has been the long-term goal of the facility, defined as producing more energy from the DT target fusion than the 3w laser energy on the target (for example, 1.9 MJ). The NIF was pursuing the ignition goal for almost 10 years, and it proved to be a scientific and engineering challenge. During 2011-2020, the target energy yield maxed at about 50 kJ, well below the ignition goal.

UNEXPECTED INTERRUPTION

The NIF's quest for ignition and scientific discovery was abruptly interrupted when in March of 2020 a "shelter-in-place" order was issued at our location to quench a spike of COVID-19 infections. Facility shifted to minimal safe operations, shots pauseda, and the Control Room staff reduced from 14 to 3 operators. Controls and IT teams have supported the change, assuring continuity of operations of the networks, hardware and ICCS in a non-shot, monitoring mode.

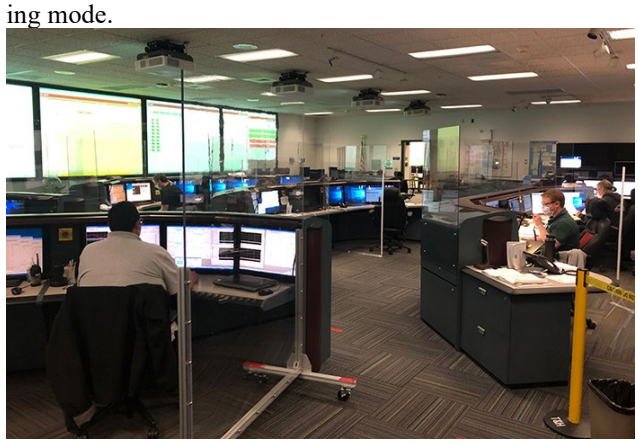


Figure 2: NIF Control with COVID-19 personnel protections: plexiglass screens, masks, traffic barriers.

Soon after the U.S. Center for Disease Control (CDC) recommendations became available, the NIF started to reorganize facility operations. Ventilation flows were adjusted, plexiglass screens between the consoles were installed and personnel traffic was directed with barriers to assure social distancing of at least 7.5ft (Fig. 2,3). Our teams have supported gradual restoration of normal shot operations, by May 2020. Quick restart of experiments was welcomed by the U.S. National Nuclear Security Administration [2,3].

While initially the restarted operations progressed slowly and deliberately, soon the shot rate had ramped up to normal, and control teams had to address the need for support and maintenance activities such as ICCS releases. Traditionally, for major software releases as well as testing and troubleshooting, ICCS software engineers were

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

present in the NIF Control Room (Fig. 4). Now, with the Control Room access limited to operator shift only, and most of the software groups and IT teams working remotely from their homes, we had to come up with new processes and tools.

Figure 3: Access to the Control Room is restricted.

Control software engineers, in collaboration with IT, have enthusiastically explored remote access and collaboration tools available on our enterprise networks.

Figure 4: Before COVID-19, ICCS software engineers can often be seen joining operators in the Control Room.

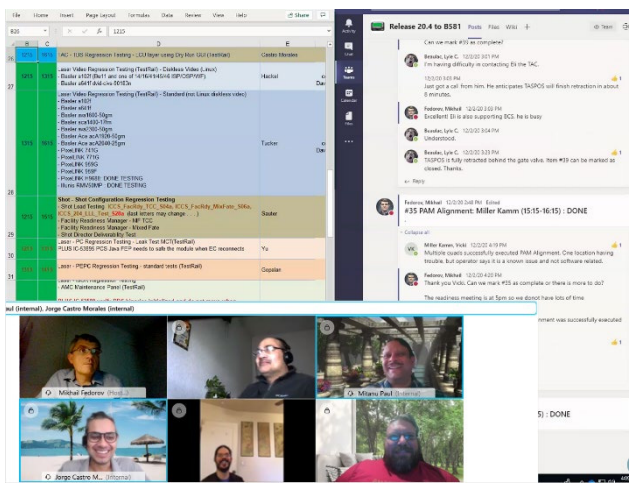


Figure 5: New all-remote software release process.

Our team has settled on Cisco WebEx for remote teleconferencing (similar to Zoom), Microsoft Teams for persistent group chat (similar to Slack) and Microsoft Office 365 for distributed "cloud" collaboration (similar to Google Docs). Both software release coordination and off-hours support have moved to these platforms (Fig. 5) [4].

EXPANDING LASER AND DIAGNOSTICS CONTROLS CAPABILITIES

With COVID-19 concerns addressed and mitigated, the control teams have returned to their primary mission of expanding and maintaining capabilities to the benefit of the NIF programmatic missions.

Target Diagnostics Controls

The NIF is well equipped with target diagnostic instruments to capture laser-target interactions in time and space using various bands of electromagnetic spectrum and particles. Over 87 of the active target diagnostics are controlled by ICCS software control system, combining over 600 rich API devices of 55 types (cameras, oscilloscopes, alignment, etc.).

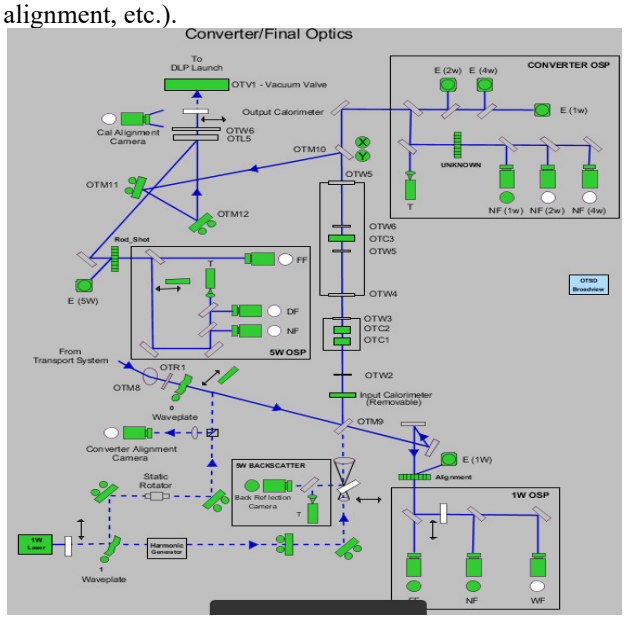


Figure 6: One of user interface panels developed for new Optical Thomson Scattering (OTS) Laser and Diagnostic.

Several new diagnostic instruments, of varying complexity, are being added to ICCS every quarter. One of the most sophisticated, the Optical Thomson Scattering (OTS), combines a high-power 5w deep ultraviolet (UV) 210 nm laser with a set of spectrometers to serve as a "plasma laser probe" (Fig. 6, 7)

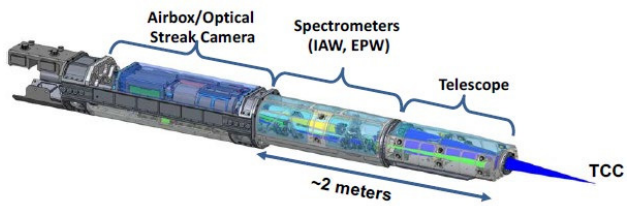


Figure 7: OTS instrument.

The design of the OTS control system and its alignment algorithms are covered by separate presentations at this conference [5,6].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL02

Laser Diagnostics Controls

While ICCS Target Diagnostics capture a variety of data about behaviors of the targets, it is equally important to precisely characterize the "driver", the 3w laser light which compresses NIF targets. Due to the design of Final Optics Assemblies (FOAs), the conversion of light from infrared 1w to ultraviolet 3w happens right at the vacuum windows where beams enter the target chamber. While a fraction of the converted 3w light is split off and delivered to ICCS diagnostics, it was a long-standing question how precisely these measurements correspond to energy of light on the target.

New Target Chamber Calorimeter (TC-CAL) was designed to obtain the energy measurements by placing a set of NIST-calibrated calorimeters into the target chamber, with a pinhole to separate 3w from unconverted 1w and 2w light (Fig. 8).

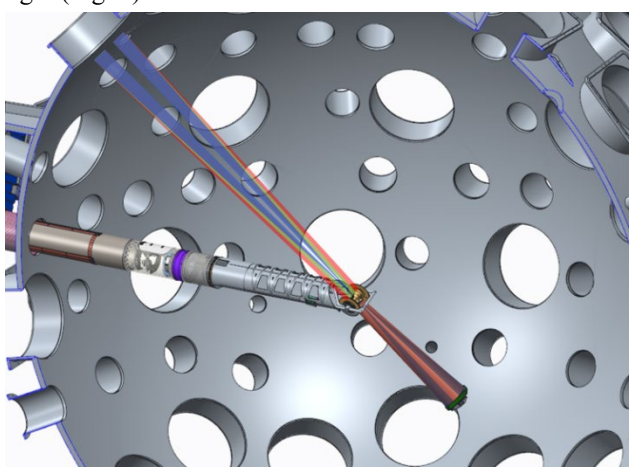


Figure 8: Target Chamber Calorimeter.

TC-CAL is a large (4.5 m+) and heavy (350 kg) mechanical device. ICCS developed a precise motion control and alignment system based on ATLAS [7], achieving excellent alignment of beams on the calorimeters, confirmed by a built-in alignment video camera.

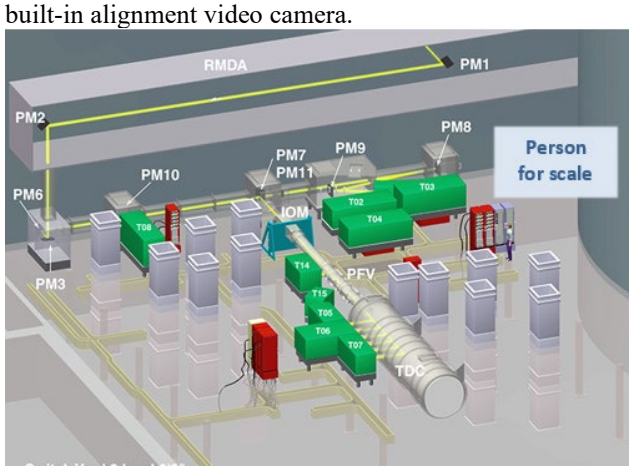


Figure 9: Layout of Precision Diagnostic System.

Another approach to study the 1w light and non-linear process of conversion to 3w is to redirect a full 40 cm x 40 cm NIF laser beam into the specialized lab, Precision Diagnostic System (PDS), (Fig. 9). PDS was recently reactivated, adding over 400 control devices. In addition, a new motorized mirror pickoff was implemented, so a selected beam is automatically routed for a detailed characterization by calorimeters, oscilloscopes, cameras, and spectrometers on multiple 1w and 3w diagnostic tables.

Expanding Laser Master Oscillator Controls

In addition to diagnostics, the laser pulse generation continued to improve as well. In ICCS architecture, the Master Oscillator Room (MOR) subsystem is responsible for the devices which generate the initial fiber laser pulse, and then shape it in time and space. For some experiments, it is desirable to have independent wavelength (or "color") control of NIF laser beam cones. In 2020, we have added a fourth master oscillator, for Outer 50 cones (Fig. 10).

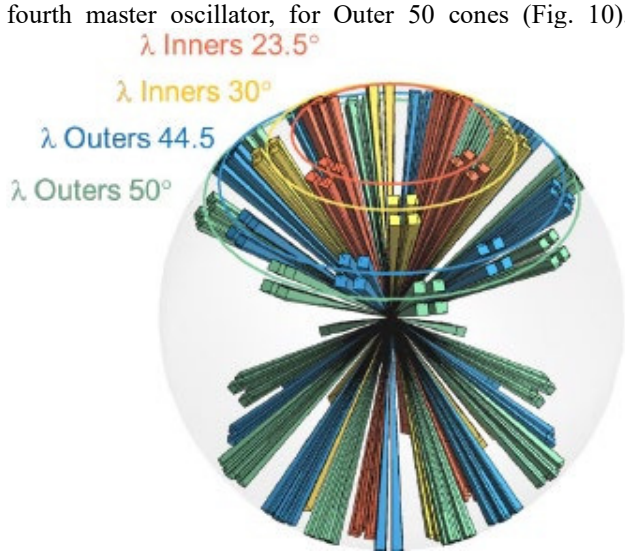


Figure 10: New dedicated laser wavelength "color" was added for Outer 50 cone of laser beams.

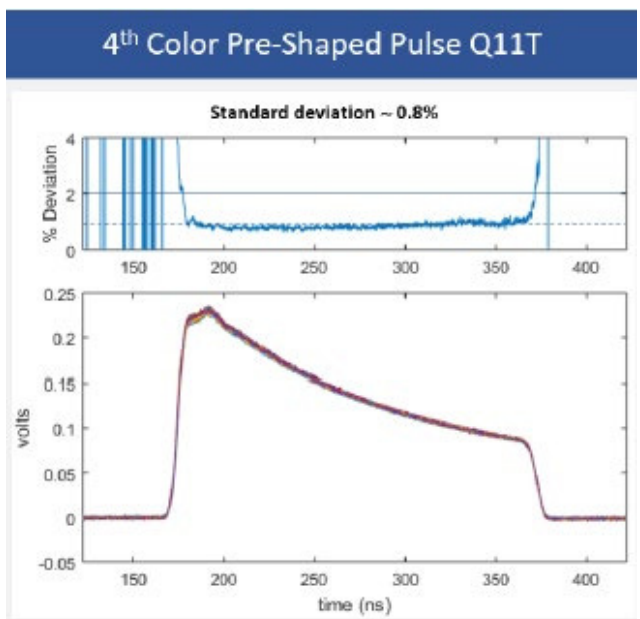


Figure 11: Laser pulse shape stability shot-to-shot improved 3X with the completion of 4th Color project.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Once the new hardware components and new enhanced design, the Pre-Shaped Pulse Generation (PSPG), were utilized, the effort also resulted in 3X improvement in shot-to-shot stability of the laser pulse shapes (Fig. 11) [8].

The precision of the laser pulses, their reproducibility and energy balance are critical for the quality of scientific data produced by NIF. These efforts will continue, with the next phase of the High-Fidelity Pulse Shaping (HiFiPS) effort starting in 2021-22. The software development work is already underway to develop low-level interfaces for new pulse shaping hardware, to fine-tune new shaping algorithms. The goal is to achieve 4X better short-term pulse shape stability, and better than 0.5% accuracy of actual waveform relative to the requested.

Recent Achievements

The multi-year efforts by NIF, LLNL and the broad scientific community to better understand and control ICF processes continue to pay off. The progress towards the ignition goal was remarkable this year, breaking both 100 kJ and 1 MJ fusion energy barriers.

The most recent record shot on August 8, 2021, produced over 1.3 MJ of fusion energy, putting researchers at the threshold of ignition (Fig. 12) [9].

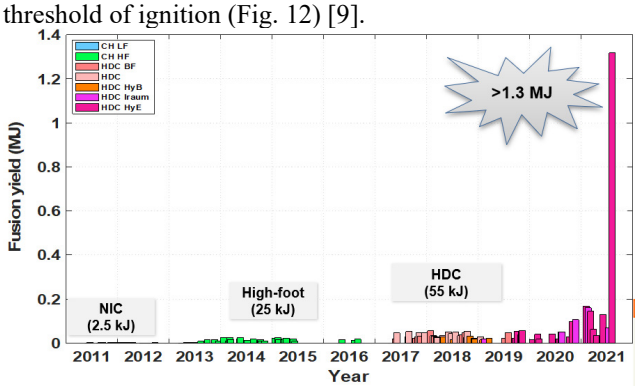


Figure 12: Recent progress toward ignition goal.

LONG-TERM SUSTAINMENT

Multiple priorities are driving control system organizations at large experimental physics facilities: assuring high 24x7 availability, continuous expansion of capabilities, and efficiency optimizations to produce even more high-quality data. And since our facilities represent significant investment of public funds and human capital, we also need to assure that facilities will continue to produce knowledge for the next 10-20 years.

Increasing uptick in failures of aging hardware, and obsolescence of computing platforms call for a broad technology refresh effort. By 2020, NIF ICCS software team has successfully completed a multi-year effort to migrate from legacy Ada 95 codebase to Java, from PowerPC VxWorks and Sun Solaris to Intel Linux – all without stopping 24x7 facility operations [10].

Our focus for 2020-2021 is to migrate hundreds of NIF video acquisition systems from proprietary Windows software to Linux, employing open-source FireWire and GigE drivers, making these systems diskless for high availability, consistency, and cyber security [11].

Looking further, we will be migrating a variety of NIF embedded controllers from aging PC104/VxWorks units to a modern, small factor platform. To support a broader range of R&D laser projects within Photon Sciences, we have proposed and continue to advance a lightweight framework for distributed LabVIEW-based control systems [12].

CONCLUSION

The NIF control systems teams: hardware, IT and software have successfully mitigated the unexpected challenge of COVID-19, adapting to "New Normal" with more health protections, social distancing, and remote work.

Broad and deep 10-year investigation of the most fundamental behaviours of laser, plasma and target interactions has started to pay off, with fusion energy outputs rising, bringing NIF to the threshold of thermonuclear ignition.

Increased neutron yields attach new urgency to already planned sustainability efforts. Some of the video camera systems in the Target Bay will need to be replaced sooner than originally planned: the Target Alignment System (TAS 2.0), Chamber Center Reference (CCRS), Final Optics Inspection (FODI).

NIF entered a new experimental regime of much higher fusion yields, requiring a new generation of time-resolved diagnostics, such as Magnetic Recoil Spectrometer (MRSt).

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Document number: LLNL-CONF-827321

REFERENCES

[1] M. L. Spaeth *et al.*, "Description of the NIF Laser," *Fusion Sci. Technol.*, vol. 69, p.25-145, Mar. 2017 http://dx.doi.org/10.13182/FST15-144

[2] COVID-19 Protocols Keep NIF on Track, https://lasers.llnl.gov/news/covid-19-protocols-keep-nif-on-track

[3] National Nuclear Security Administration. Experiments resume at the National Ignition Facility May 8, 2020, https://www.energy.gov/nnsa/articles/experiments-resume-nation al-ignition-facility

[4] Remote NIF Control Software Updates Lead to Less Stress, More Efficiency, https://lasers.llnl.gov/news/remote-nif-control-system-software up dates-less-stress-more-efficiency

[5] A. Barnes *et al.*, "Upgrading the National Ignition Facility's (NIF) Integrated Computer Control System to Support Optical Thompson Scattering (OTS) Diagnostic", presented at ICALEPCS2021, Shanghai, PRC, Oct. 2021, this conference.

[6] A. Awwal *et al.*, "Image Processing Alignment Algorithms for the Optical Thomson Scattering Laser at the National Ignition Facility", presented at ICALEPCS2021, Shanghai, PRC, Oct. 2021, this conference.

[7] R. Wilson *et al.*, "Experiences with Laser Survey Instrument Based Approach to National Ignition Facility Diagnostic

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL02

Alignments", presented at ICALEPCS2017, Barcelona, Spain, October 2017.

[8] Di Nicola, J. M., D. Kalantar, S. T. Yang, D. Alessi, T. Bond, M. Bowers, B. Buckley *et al*., The National Ignition Facility Laser Performance Status. No. LLNL-PROC-819559. Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2021.

[9] NIF Experiment Puts Researchers at Threshold of Fusion Ignition
https://lasers.llnl.gov/news/nif-experiment-puts-researchers thres hold-fusion-ignition

[10] M. Fedorov *et al*., "In-Place Technology Replacement of a 24x7 Operational Facility: Key Lessons Learned and Success Strategies from the NIF Control System Modernization", presented at 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019), New York, U.S.A, October 2019.

[11] V. Gopalan *et al*., "Modernizing Digital Video Systems at the National Ignition Facility (NIF): Success Stories, Open Challenges and Future Directions", presented at 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, PRC, Oct. 2021, this conference.

[12] B. Davis *et al*., "Photon Science Controls: A Flexible and Distributed LabVIEW Framework for Laser Systems", presented at 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, PRC, October 2021, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

FROM SKA TO SKAO: EARLY PROGRESS IN THE SKAO CONSTRUCTION

J. Santander-Vela[1*], M. Bartolini[1,2], M. Miccolis[1], N. Rees[1]
[1]SKA Observatory, SK11 9FT Jodrell Bank, United Kingdom
[2]INAF Istituto Nazionale di Astrofisica, Viale del Parco Mellini 84, 00136 Roma, Italy

Abstract

The Square Kilometre Array telescopes have recently started their construction phase, after years of pre-construction effort. The new SKA Observatory (SKAO) intergovernmental organisation has been created, and the start of construction (T_0) has already happened. In this talk, we summarise the construction progress of our facility, and the role that agile software development and open-source collaboration, and in particular the development of our TANGO-based control system, is playing.

INTRODUCTION

The Square Kilometre Array (SKA) is an international project that has the aim of building two multi-purpose radio telescope arrays. One of them will be built in South Africa in the Karoo desert, and the other will be constructed in the Murchison Shire in Western Australia. The name comes from the initial intention for these telescopes to provide the equivalent collecting area of at least one square kilometre, and thus unprecedented sensitivity, which would allow key questions in modern astrophysics and cosmology to be answered.

The original *Hydrogen Array* concept of an array that was sensitive enough through a very large collecting area of up to one square kilometre was described by Peter Wilkinson in 1991 [1]. One of the main concepts took the name of the Square Kilometre Array project, and several milestones were achieved in order to make this project a reality.

After several forms (from an interest group to the International SKA Project Office, later the SKA Project Office in Manchester University), and several EU framework programs (SKADS, the SKA Design Study; PrepSKA, preparation for SKA), the SKA Organisation was founded in November 2011 as a non-for-profit limited responsibility company established in England and Wales.

As part of SKADS, the first SKA Science book was published in 2004 [2]. After the official start of the SKA Pre-Construction in 2014, an update to the SKA science book was published [3] in 2015 after a decade of development of the SKA concept, incorporating more than 130 scientific use cases that will be possible thanks to the SKA telescopes.

Those science cases cover Galaxy Evolution, Cosmology and Dark Energy[1], Strong-Field Tests of Gravity[2], Cosmic Magnetism[3], The Cosmic Dawn and the Epoch of Reionisation[4], and research on the Cradle of Life[5]. The amount of physical disciplines foreseen to be encompassed by the SKA telescopes is one of the largest for any ground based facility to date.

The SKA project is currently in what is known as SKA Phase 1, or SKA1, in which two telescopes approximately with 10% of the target collecting area are being built, namely SKA1-Mid, and SKA1-Low, in order to prove the feasibility of the techniques and derisk the construction of the next phase of the project, SKA Phase 2, or SKA2.

The goal is to have a single observatory entity, that will construct and operate two SKA1 telescopes (SKA1-Mid and SKA1-Low), with presence in three sites: Australia (SKA1-Low), South Africa (SKA1-Mid), and United Kingdom (Headquarters and central operations).

This talk focuses on the progress and status of the SKA project from our last status report [4] in ICALECPS'17. It starts by describing how we have migrated from the SKA Organisation and pre-construction towards the SKA Observatory (SKAO) in From SKA to SKAO and Start of Construction. We later indicate the role of software in the SKA project in Software in the SKA Project, and we provide an update on the status of our efforts in Current Status. We continue by describing the difficulties that we have been facing up to the start of construction in Challenges, and we describe future work in Next Steps, with some short Conclusions at the end.

FROM SKA TO SKAO AND START OF CONSTRUCTION

As indicated in Sec. Introduction, the Hydrogen Array concept was first published in 1991. Several studies were made to come with a concept for the realisation of that square kilometre array, and in 2008 the EU Framework Programme called SKA Design Studies (SKADS) was started, and then followed by PrepSKA. After PrepSKA, the SKA Organisation was founded as a non-for-profit, limited liability company registered in England and Wales, but it also set in motion the process for finding what would be the ultimate legal form for the SKA Observatory (SKAO), and it was finally decided that an Inter-Governmental Organisation (IGO) was the way to go.

After PrepSKA, which set up the first contacts towards the IGO, the first round of negotiations towards the SKAO IGO took place in October 2015. Several rounds of negotiations

* juande.santander-vela@skao.int
[1] http://skatelescope.org/galaxyevolution/
[2] http://skatelescope.org/gravity-einstein-pulsars/
[3] http://skatelescope.org/magnetism/
[4] http://skatelescope.org/cosmicdawn/
[5] http://skatelescope.org/cradle-life/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

where required, and they culminated with the signature of the SKAO Treaty in Rome in March 2019.

For the treaty to enter into force, it was required that the three site countries (UK, Australia, and South Africa) had to ratify the treaty, plus two more countries. The process was finally complete with the ratification of the Treaty in December 2020, and the entry into force of the Treaty happened in January 15th, 2021.

After that, the first SKAO Council met on February 4th, 2021, setting up the way for the acquisition of the assets of the SKA Organination, with personal transferred through the TUPE process. All company assets and staff were transfered to the IGO in May 1st, 2021.

All of this process can be seen in detail in Fig. 1.

In June 2021 the SKA1 Construction Proposal (CP), and the SKA Observatory Establishment and Delivery Plan (OEDP) were submitted by the SKAO Director-General with a recommendation for approval by the Council. The CP describes the science requirements for the project; what the scope of the project is; how the project will be executed, monitored, and controlled; wider benefits of the project to society; and a guide to the detailed project documentation, reference information, and its organisation; whereas the OEDP details staffing and costs for the SKA Observatory own delivery and supporting functions: Business enabling functions, Observatory operations, Observatory development, and Construction support. The Council approved both documents, and hence the start of construction (T_0) took place on July 1st, 2021.

SOFTWARE IN THE SKA PROJECT

Software plays a major role in the SKA project. A notional data flow view of the two telescopes is shown in Fig. 2. In there, some of the major blocks of the telescope requiring software are displayed:

- The Science Data Processor for both telescopes includes the real-time ingest of the data being generated by the whole of the array. This is a big-data stream of up to 9 Terabits per second, and apart from the ingest of the bulk data and its relevant metadata, some real-time calibrations need to be executed in pseudo-realtime. Moreover, the system uses batch processing on those datasets to produce science-ready data for the network of SKA Regional Centres (SRCs) to proceed and deliver to our Principal Investigators and Co-Investigators.

- The SRCs themselves are supercomputing facilities that need to be able to perform High-Performance Computing analysis on demand from the community of SKA researchers.

- The Central Signal Processor are supercomputers in their own right, but instead of being general purpose they are mostly concerned with floating-point Multiply and Accumulate operations in support of Fast-Fourier Transform of the data, implemented as Field Programmable Gate Arrays (FPGAs). The firmware/bitimages that configure those FPGAs are also being managed through the same software processes as the rest of our software.

In addition, two more large systems are software systems: the Telescope Management and Control (TMC) system is the control software that orchestrates and manages our telescopes, and is based on TANGO; and the Observatory and Science Operations (OSO) systems is the interface for researchers to submit observation proposals, and then convert them into the Scheduling Blocks that will be run by the TMC in order to carry out the observations.

CURRENT STATUS

As indicated in Sec. From SKA to SKAO and Start of Construction, the Construction phase has just started, but for software – including the SKA telescopes' control system – the development started in August 2018, using the bridging phase between pre-construction and construction to derisk our development processes and code. From that time until July 1st, 2021 we have gone already through ten full Program Increments, and the first PI to be partially in the construction phase was PI11, with PI12 – our current PI as of the writing of this article – being the first one fully within the SKA Phase 1 (SKA1) construction phase.

We are using the Scaled Agile Framework[6] (SAFe®) as the version of agile to be scaled to the needs of the SKA1 construction project. SAFe groups agile teams into Agile Release Trains (ARTs). Each ART is built around a stream of value to be delivered. Initially, we started with a single train, which delivered all software required for the SKA telescopes. When the number of people and teams increased, we moved to two trains: one of them delivering the value for the Science Data Handling and Processing (SDH&P) part, and another one for Observation Management and Controls (and Correlation; OMC). In PI11, we soft-launched a third ART, the Services ART, which provides value to both the DP and OMC ARTs in the form of System services (development platform, versioning, continous integration and continuous delivery), Platform services (definition and furnishing of the middleware that helps run all software, such as Kubernetes or Open Stack), and Networking services.

Key to the success of our software developments is providing a joined up Vision of what we want to deliver. And we update that Vision to the nearest horizon that is useful for our teams. There are 5 main milestones for the SKA telescope development, that we call Array Assemblies (AAs). In our initial roll-out plan, we had four AAs (AA1 to AA4). However, it was decided that an earlier integration testing point should be considered, and we call that milestone AA0.5.

Table 1 provides information on the different AA milestones, together with the capabilities that we expect to be

[6] https://scaledagileframework.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

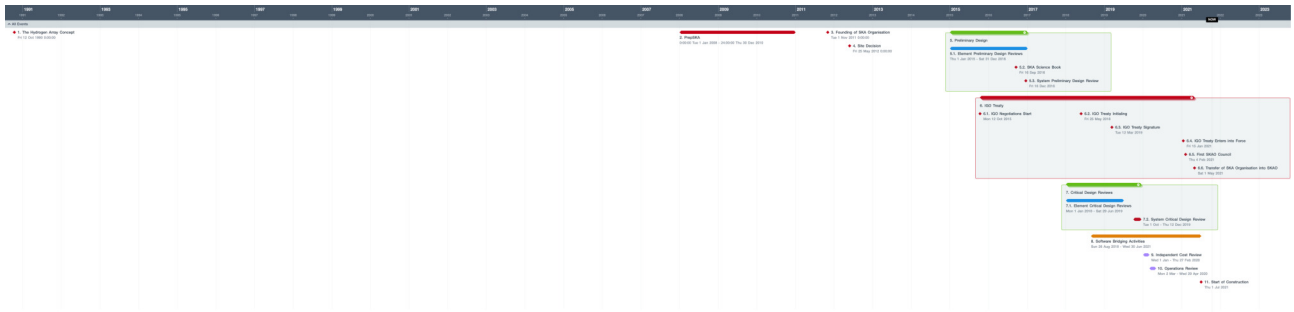


Figure 1: Timeline of the SKA project pre-construction phase. The earliest milestone took place in 1991 with the first concept of *The Hydrogen Array*. The boxes in green represent the design phases, while the items in read represent progress towards the establishment of the SKA project and the SKAO IGO. The final date for the pre-construction phase is T_0, the Start of Construction.

able to test for each one, and when we currently expect to finish those milestones.

Table 1: Sensing elements per Array Assembly

	Sensing Elements	
Milestone	**Mid**	**Low**
AA0.5	4	6
AA1	8	16
AA2	64	64
AA3	121	256
AA4	197	512

In the same vein of early testing, we started developing and planning the SKA software with our first Program Increment (PI1) in December 2018. The decision was to create a Minimum Viable Product (MVP) in the sense of Eric Rees [5]: the minimum set of features that can solve a problem and that allows learning from early adopters to be incorported. In this case, our MVP is for the first integration prototype, and we named it the SKA MVP Product Integration, or SKAMPI.

An important part of SKAMPI is the support of the Continuous Integration/Continuous Deployment pipelines in order to test and verify our software. The work done so far is summarised by Matteo Di Carlo [6] in this conference.

The development of SKAMPI in paraticular, and of all SKA software in general, needs to address the needs of each of the AAs indicated in Table 1. To indicate what needs to be done in each moment, we have developed a Solution-level roadmap, which is updated at least every PI. And starting in PI11 we are also developing a joint TDT-level roadmap, that is then interpreted at software level. However, we have not yet managed to provide that joined up TDT/Solution-level roadmap view. The closest we have is the Solution level roadmap that can be seen in Fig. 3.

That roadmap follows three main development streams:

- **SKAMPI development:** This stream includes all development effort required to provide working, useful SKAMPI software at each stage.
- **SKAMPI integration support:** This stream comprises development effort required to help the Assembly, Integration, and Verification (AIV) teams to test the SKA hardware and software.
- **Architectural Runway:** This final stream includes all development effort required to incrementally change the architecture of the software to address the needs of each of the individual AAs.

As indicated at the beginning of this section, we are currently executing PI12, and preparing the backlog and features to be planned in PI13, to be held in December 2021.

CHALLENGES

During the pre-construction PIs, and in this first construction PI, we have faced several challenges:

- Using TANGO in an event-oriented way
- Containarisation and Orchestration
- Computing Scaling
- Team Scaling

Using TANGO in an Event-Oriented Way

Most facilities using TANGO for control rely heavily on polling, as they control hardware systems for which it is expensive to obtain many of the monitoring values that might be required.

For SKA software, many of the controlled entities are other TANGO DeviceServers (DSs), and below those DSs additional software exists, which can easily change and/or generate values as required. In those cases, it is preferred to have an event-based system, in which data is only updated when an change-event is required.

Most SKA software has moved away from polling, and this has found some edge cases around the TANGO event mechanism, and the need for more TANGO expertise, and potentially the ability to contribute to change the TANGO event engine is required.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

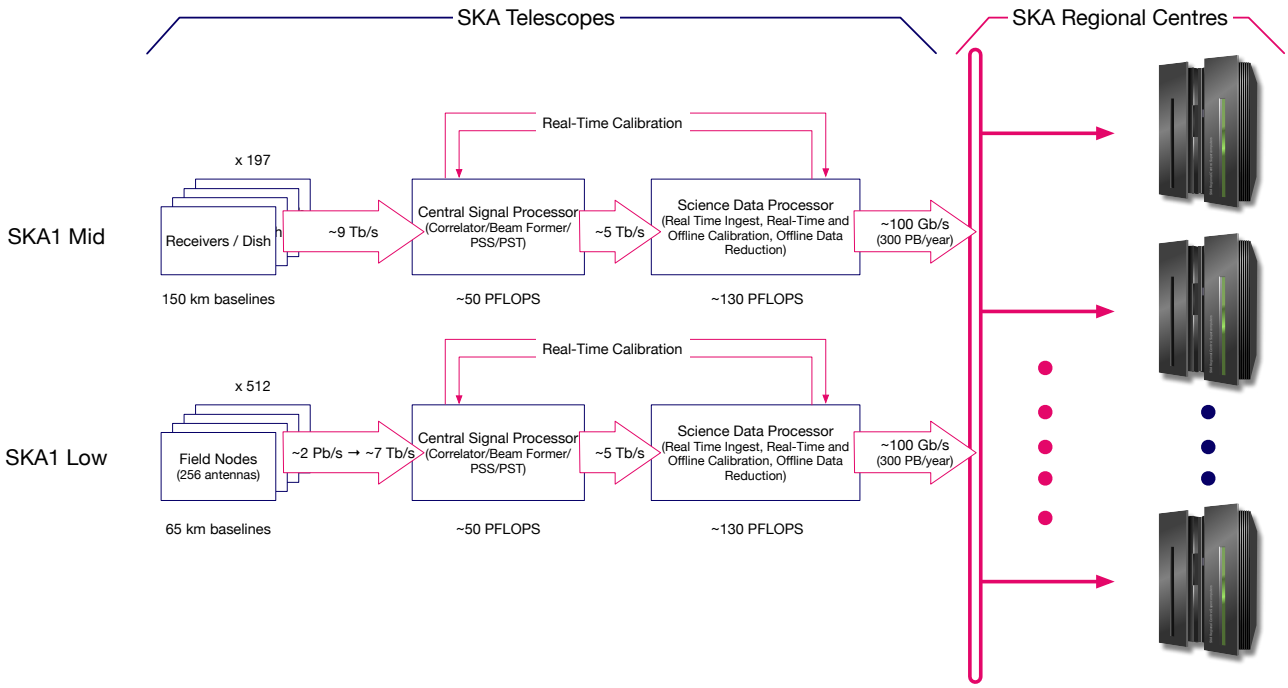


Figure 2: Notional data flow of the two SKA1 telescopes, SKA1-Mid and SKA1-Low. The different maximum bandwidths for each flow are shown in each arrow. The data generated by the telescopes is distributed to the network of SKA Regional Centres which are not in the scope of the SKA1 construction project.

There are additional issues coming from the interaction of the event system with containarisation and orchestration, but we will discuss those in the next subsection.

To try to resolve those issues, we have hired a new member of staff at SKAO that will contribute 50% of their time to support the TANGO kernel, while providing additional TANGO development support to the teams. This is part of SKAO's contribution to the TANGO Collaboration, since we joined in October 2017.

We have also developed ways to log and trace events, as indicated by Samuel Twum [7] in this conference.

Containarisation and Orchestration

In order to be able to scale the execution and deployment of SKA software, and to orchestrate both its deployment and runtime environment, we are using Open Container Initiative (OCI) containers[7] – although we started with Docker[8] containers. For orchestration, we started using Docker Compose[9], but we have recetly moved to use Kubernetes[10] (K8s), and Helm[11] charts for managing all packaing, deployment, and software sequencing.

Figure 4 shows the different labeled groupings of software, and how they are mapped into services, pods, and containers. Currently, a single TANGO DS leaves inside a container.

[7] urlhttps://opencontainers.org
[8] http://docker.com
[9] https://docs.docker.com/compose/
[10] http://kubernetes.io
[11] https://helm.sh

We are currently being challenged by the learning curve of K8s and Helm for some of our developers. This is another instance in which having an standardise approach within the TANGO community will be beneficial, and SKAO would like to foster such an outcome.

Computing Scaling

As indicated in the previous subsection, we are currently using a 1:1 mapping between containers and TANGO DS, but also a 1:1 mapping between DSs and TANGO Devices. However, that granularity seems to impose a high-overhead, and we are looking into how to better combine DSs within containers, and Devices within Device Servers.

We are also finding some issues in how to express some of the dependencies, and linkages, through the Helm charts, to that we can properly map software releases, and combine them into a single product.

Team Scaling

The last challenge has to do with how our software development processes are scaling.

A large infrastructure project such as the construction of the SKA telescopes requires strong project management skills, and fixing a number of milestones to ensure proper monitoring of the progress. There is an Integrated Product Schedule (IPS) which is the source of truth for the major project milestones, and includes both schedule and costing information.

Converting that IPS into actionable information for the software development teams requires a lot of bandwidth

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

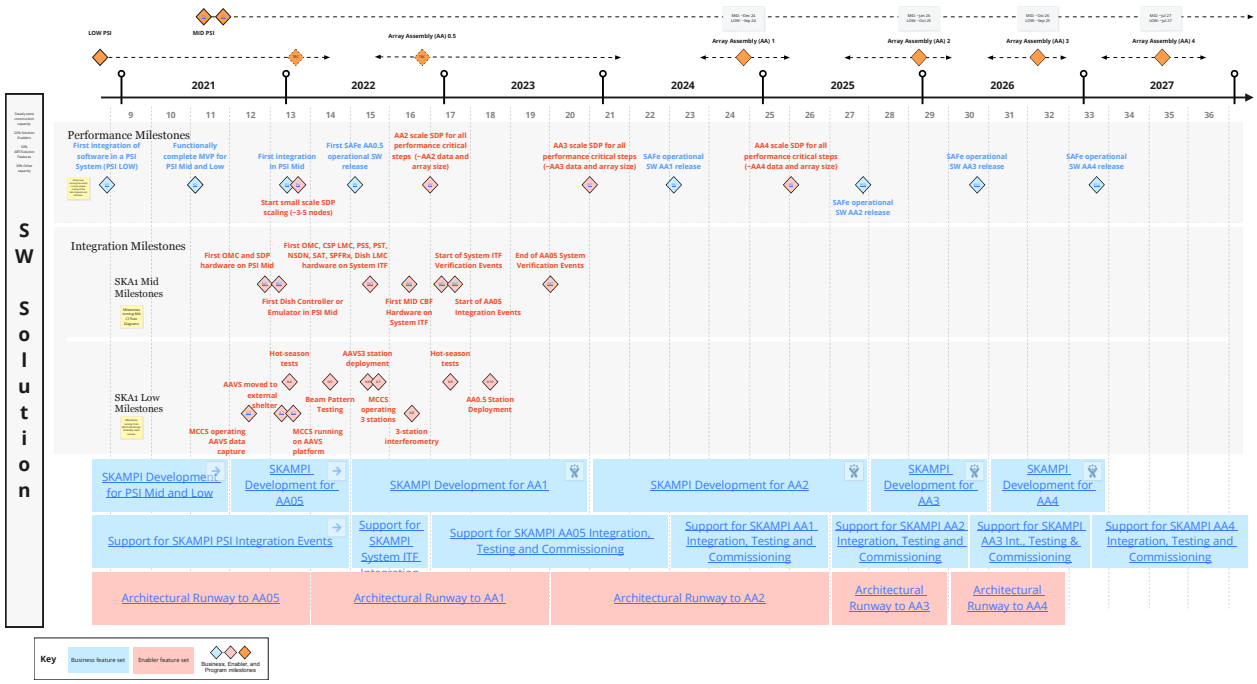


Figure 3: Long term Software Solution roadmap for all SKAO construction, and part of the pre-construction. Each Program Increment (PI) corresponds to 13 weeks, so there are exactly 4 PIs per year, and each PI planning event tends to happen within the same week every year, which helps with the predictability for the dates. The rhombs above the years corresponds to milestones that have to do with the telescope roll-out plans: PSIs stand for Prototype System Ingegrations. AA stands for Array Assembly, and each Array Assembly corresponds to increased number of equipment delivered, and additional capabilities. Performance Milestones have to do with when particular levels of performance, or non-functional requirements (NFRs), need to be met, in order to achieve the right level of performance when delivering the final system. Integration Milestones, on the other hand, indicate when particular functionality of the software needs to be available because of integration and testing requirements. The bottom row corresponds to three lanes of work: a) development stream of the SKA Minimum-viable-product Prototype Integration (SKAMPI); b) support stream of the usage of SKAMPI in several environments, including the AAs; and c) changes and evolution required in the software architecture related to the requirements of the different AAs.

between the TDTs and the software teams, and is something that we are working on to improve.

We are also starting to develop better rules for contribution and release management, that we expect to have in place soon.

NEXT STEPS

In order to address the previously outlined challenges, we are working to improve the communication and aligment between the IPS and the Software Roadmap: we start of TDT Planning as part of PI planning is helping with that, and we have also started to run dedicated roadmap alignment workshops to make sure that all dependencies and team needs are surfaced well ahead of PI planning.

We are also finishing with the qualification of suppliers, so that we can start proper contracting under construction funds, and finally move from bridging to construction funding.

Some of the issues with both Team Scaling and Computing Scaling have to do with lack of appropriate funding of effort to the System and Platform teams. We exected that setting up the contracting framework should help with this, by both accessing *in-kind*, and commercial contributors to these teams.

Finally, we will keep iterating on our TANGO base software (as described by Sonja Vrcic [8] in this same conference), and keep iteraring our MVP of Product Integration in order to move to the first releases of proper SKAO Telescope Software.

CONCLUSIONS

The construction phase that will deliver the transformational SKA telescopes has just started. After passing T_0 in July 1st, 2021, we have already issued several contracts for construction (C_0), but have also already been taking advantage of bridging funding in order to kickstart the development of the software. We have found some challenges, but we are doing so at the very start of the project, which makes it easier to resolve. We expect to be able to deliver our software on time, and to the satisfaction of our customers,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

which are primarily the AIV and Commissioning and Science Verification (CSV) teams.

ACKNOWLEDGMENTS

J. Santander-Vela wants to dedicate this paper to the memory of his recently passed-away father-in-law, Mr. Demetrio Sababa. He also wishes to acknowledge the work of all SKAO engineers, and the support of the co-authors.

REFERENCES

[1] P. N. Wilkinson, "The Hydrogen Array," in *IAU Colloq. 131: Radio Interferometry. Theory, Techniques, and Applications*, T. J. Cornwell and R. A. Perley, Eds., ser. Astronomical Society of the Pacific Conference Series, vol. 19, Jan. 1991, pp. 428–432.

[2] C. L. Carilli and S. Rawlings, "Science with the Square Kilometer Array: Motivation, key science projects, standards and assumptions," *New Astronomy Review*, vol. 48, no. 11-12, pp. 979–984, Dec. 2004. DOI: 10.1016/j.newar.2004.09.001. arXiv: astro-ph/0409274 [astro-ph].

[3] *Advancing Astrophysics with the Square Kilometre Array*, Apr. 2015.

[4] J. Santander-Vela, L. Pivetta, and N. Rees, "Status of the Square Kilometre Array," in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017*, (Barcelona, Spain), ser. International Conference on Accelerator and Large Experimental Control Systems, https://doi.org/10.18429/JACoW-ICALEPCS2017-FRAPL01, Geneva, Switzerland: JACoW, Jan. 2018, pp. 1982–1988, ISBN: 978-3-95450-193-9. DOI: https://doi.org/10.18429/JACoW-ICALEPCS2017-FRAPL01. http://jacow.org/icalepcs2017/papers/frapl01.pdf

[5] E. Rees. "Minimum viable product: A guide." (Aug. 2009), http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

[6] M. Di Carlo, "Continuous integration-continuous delivery (ci-cd) practices at ska," presented at ICALEPCS'21, Shanghai, China, 2021, paper TUBL04, this conference.

[7] S. Twum, "Implementing an event tracing solution with consistently formatted logs for the ska telescope control system," presented at ICALEPCS'21, Shanghai, China, 2021, paper TUBL02, this conference.

[8] S. Vrcic, "Design patterns for the ska control system," presented at ICALEPCS'21, Shanghai, China, 2021, paper TUBR02, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

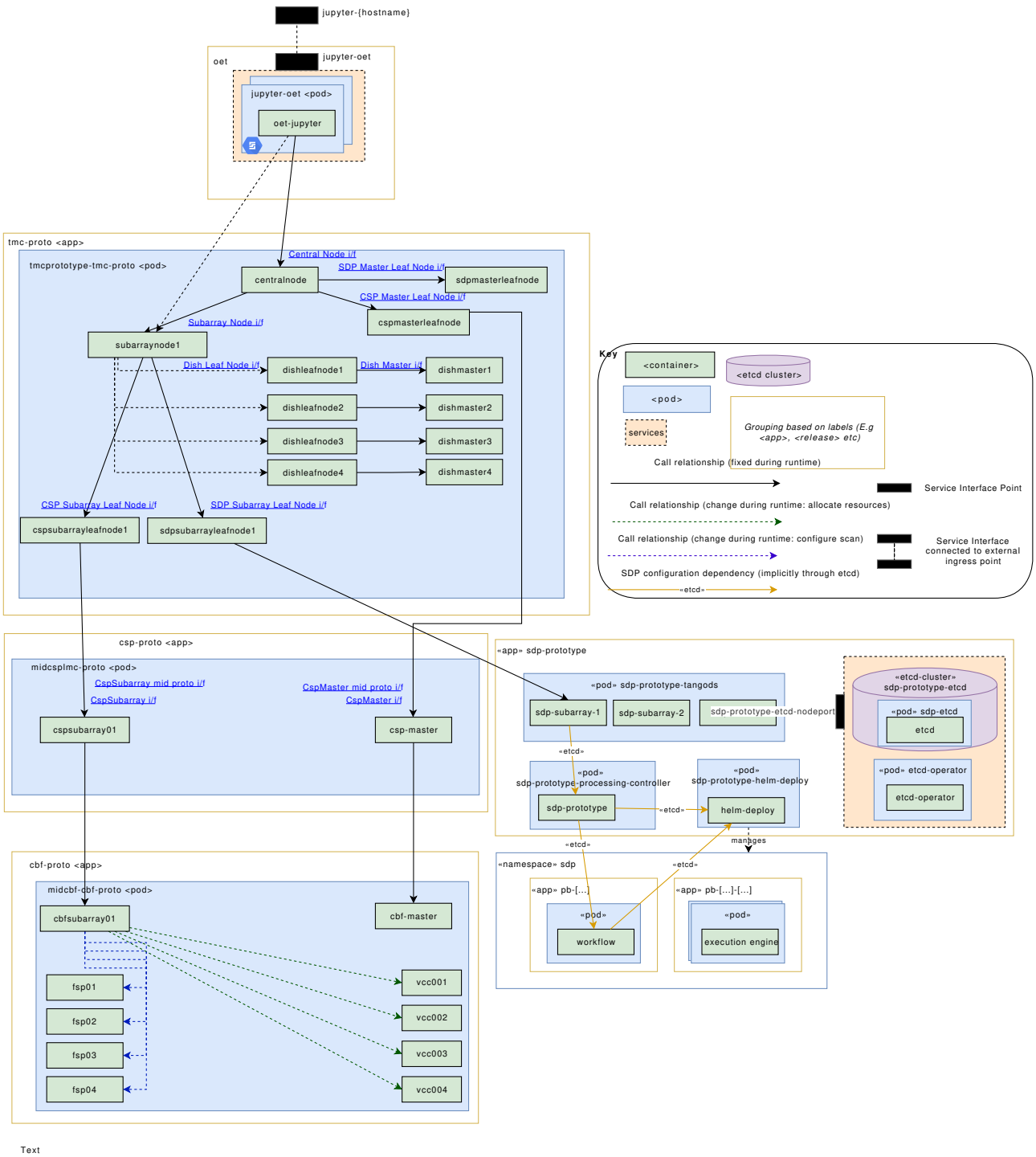


Figure 4: Connector and Component view of SKAMPI software.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR01

MODERNIZING THE SNS CONTROL SYSTEM*

K. S. White†, K. Kasemir, K. Mahoney, K. Vodopivec, D. Williams,
Oak Ridge National Laboratory, Oak Ridge, USA

Abstract

The Spallation Neutron Source at Oak Ridge National Laboratory has been operating since 2006. An upgrade to double the machine power from 1.4 MW to 2.8 MW is currently underway and a project to add a second target station is in the preliminary design phase. While each project will add the controls needed for their specific scope, the existing control system hardware, software, and infrastructure require upgrades to maintain high availability and ensure the system will meet facility requirements into the future. While some systems have received new hardware due to obsolescence, much of the system is original apart from some maintenance and technology refresh. Software will also become obsolete and must be upgraded for sustainability. Further, requirements for system capacity can be expected to increase as more subsystems upgrade to smarter devices capable of higher data rates. This paper covers planned improvements to the integrated control system with a focus on reliability, sustainability, and future capability.

BACKGROUND

The Spallation Neutron Source (SNS) is an accelerator-based neutron facility that provides the world's most intense source of pulsed neutrons for research. The machine was originally commissioned in 2006 and began operating for users in 2007. The facility was constructed by a collaboration of six laboratories who delivered controls along with their systems. The SNS Controls Group was responsible for global systems, control system infrastructure and integration of the partner contributions to create a single Integrated Control System (ICS) [1]. Hardware and software standards were adopted for the control system including the selection of the Experimental Physics and Industrial Control System (EPICS) toolkit for control, communication, and operational services. Using EPICS then allowed a diverse set of hardware to be integrated into a cohesive system.

The client/server architecture of the SNS control system consists of three layers shown in Fig. 1. The front-end layer employs input/output controllers (IOCs) as servers to connect to devices in the field and execute run-time control functions. The communication layer passes data in the form of EPICS process variables (PVs) between front-end IOCs and client applications. The back-end layer uses workstations to execute client applications to provide operational interfaces and tools.

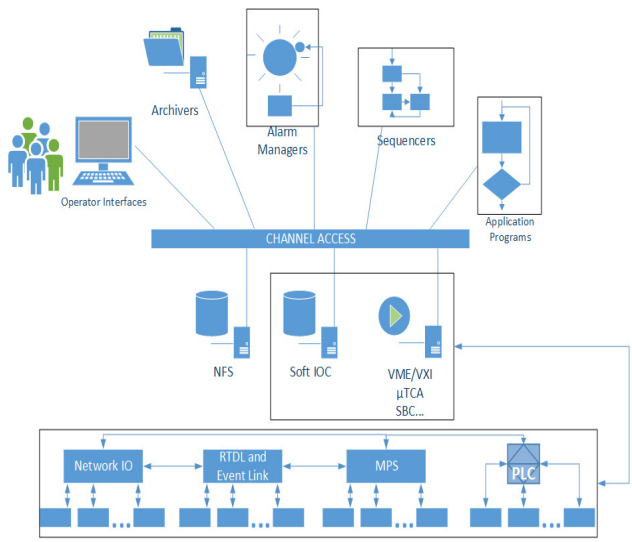


Figure 1: SNS ICS Architecture.

A key advantage of this type of architecture, where control functions, communications, and operational tools are decoupled, combined with the inherently distributed design of EPICS, is the ability to upgrade the layers, or individual components of a layer, independently thus providing a minimally disruptive path for upgrades which is essential for operating facilities.

The ICS, as originally constructed, employed commercial Linux servers and workstations for client layer and standard ethernet networking for communication. EPICS client applications providing operator interfaces included a custom alarm handler, the Extensible Display Manager (EDM) and the Channel Archiver. Several different types of IOCs were used, including VME/VXI Motorola single board computers running VxWorks, Allen Bradley Programmable Logic Controllers (PLCs) interfaced to Linux based (soft) IOCs or VME IOCs, and Windows PCs running Labview for Beam Instrumentation Devices. Table 1 shows the number of IOCs, PLCs, and EPICS PVs during initial operations and the present, pointing out where the system has expanded. While the number of VME IOCs has remained about the same, the number of soft IOCs has tripled supporting requests for new features as well as new devices and subsystems and the number of PVs has grown by ~50%.

* Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).
† ksw@ornl.gov

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 1: ICS Growth

	2006	2021
VME IOCs	168	167
Linux IOCs	46	152
µTCA	0	9
Windows IOCs	248	170
PLCs	100	189
MPS inputs	923	1000
PVs	395000	603000

This combination of hardware and software, integrated using EPICS, provided a reliable control system, built on a foundation that could be extended by adding new IOCs and applications. However, it was recognized during early operations that improvements to the alarm handler would be necessary and some premature PLC hardware failures pointed to manufacturing defects that led to early upgrades of these processors. The availability history of the ICS is shown in Fig. 2. After an initial operations period where many problems were addressed with many systems, the original control system design and architecture are meeting the needs of the facility, however, many system components are aging and without upgrades, the ability to maintain high availability and support additional devices for new systems will be increasingly challenged.

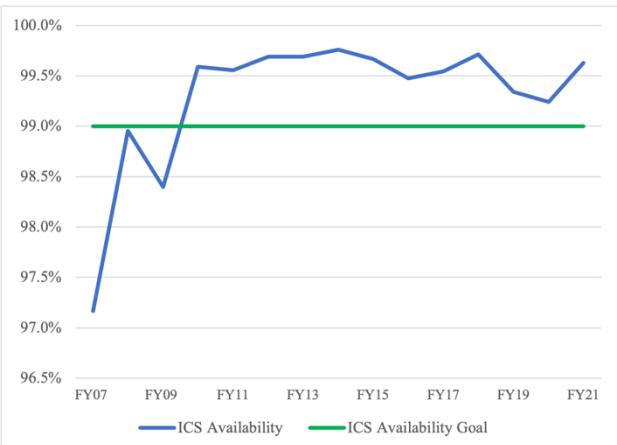


Figure 2: ICS Operational Availability.

LOOKING FORWARD

Over time, experience with the control system and performance analysis has informed planning for various upgrades. Predictably, hardware failure rates increase with years of operation and as parts become obsolete, the ability to repair failed hardware erodes. While not failing in the same way as hardware, software and firmware become compromised over time as vendors withdraw support for older versions or operating system upgrades are required, often to maintain an essential level of cyber security. Eventually, without upgrades, it becomes impossible to maintain platforms capable of compiling software or the tools needed to generate executable files creating an untenable situation for machine support and ruling out upgrades or any changes to system functionality. As smarter field devices are introduced due to upgrades of other systems the control system must also be prepared to support more modern hardware, often capable of higher data rates than original systems. The SNS also has two facility upgrade projects underway which will require additional control system capability.

The SNS facility has evolved since initial low power commissioning to run the full design beam power of 1.4 MW with > 90% availability. The initial suite of four instrument beamlines has been expanded to twenty and there are two construction projects underway that will expand the facility capabilities [2]. The Proton Power Upgrade project, scheduled to be completed in 2024, will increase the machine power from 1.4 MW to 2.8 MW by adding seven cryomodules in the Linac, replacing some Ring magnets and power supplies and upgrading the target design along with some elements of conventional and target utilities. The ICS will be extended to provide controls for the additional and upgraded systems.

The Second Target Station (STS), also in progress, will use the increased beam power to provide neutrons in an additional target station, initially with eight instrument beamlines and space for fourteen more. To create neutrons for the new target station, the 60 Hz proton beam will be shared between the existing target and the new one, typically by delivering 45 pulses to the first target station and 15 pulses to the second. The accelerator control system will be extended to new devices installed to create a new beam transport line. Further, the ability to provide independent operation of beam power to two destinations will require modifications to the timing system, machine protection system (MPS), kickers and the low-level RF (LLRF).

UPGRADES

Due to the flexibility and scalability of EPICS, upgrades can be pursued in parallel for the various control system layers and components while keeping the same underlying system architecture. While the primary motivation for most upgrades is to ensure the control system remains sustainable and capable of high availability operations and extensible such as is needed by the PPU project, the STS project will require a new capability to split the beam pulses between two destinations at different power levels.

Global Systems

The global systems in the SNS control system include the timing system, MPS and EPICS software along with the network and computing infrastructure. These systems can be complicated to upgrade due to their interaction with most devices on the controls network.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR01

Timing The timing system generates and distributes timing events as well as real-time data for each 60 Hz machine cycle to accelerator and instrument systems to synchronize system operations across the complex [3]. The original timing system, developed by partner labs, was based on the design and hardware of the existing Relativistic Heavy Ion Collider timing system at Brookhaven National Laboratory. The timing master was implemented using a collection of commercial and custom VME cards along with a complex real-time software application. The master communicated to a distributed network of fanout and receiver cards to deliver triggers and events.

While this system implementation performed the required functions, the installed hardware was relatively old technology and parts obsolescence became an issue after only a few years of operations [4]. Goals for an upgrade to this system included reducing the system complexity, improving availability and sustainability, and providing a phased upgrade path that would allow a gradual replacement of obsolete hardware without an extended machine outage period. Three custom boards were designed to replace a variety of VME boards, and the new design moved most of the previous software functionality into an FPGA. The new fan-out and timing receiver hardware was phased in during maintenance outages over several years and the timing master was commissioned and turned over to operations in October 2019. In addition to the original event and real-time data distribution based on twinaxial copper cabling, the new fan-outs also support a more convenient single-link fiber for newer timing receiver hardware. The firmware for the new timing system will be modified to provide the appropriate information, such as pulse destination, to hardware subsystems to support two beam operations for STS.

Machine Protection The machine protection system is designed to trip the beam within 20 µsec in the event an off normal condition is detected to prevent damage to the machine. This system is currently being upgraded due to sustainability issues with legacy hardware components that have reached end-of-life, and for which limited spares are available. The new design for this system utilizes commercial µTCA hardware with field replaceable units to reduce mean time to repair and minimize the number of custom interface components and therefore the probability of failure. The system consists of one master unit and ~64 field nodes connected to ~1000 devices which can signal the need to shut down the beam. The new MPS hardware has been installed and corresponding firmware and software have been developed to support witness testing during the run cycle in early 2022. For this test, the system will collect data without shutting down the machine, to verify it provides the same protection as the original system before the system is phased into machine operations. New MPS nodes for the PPU project will be built with the new MPS hardware.

EPICS The EPICS toolkit is used throughout the ICS to provide processing of control algorithms, communications, and operator interfaces. EPICS software is written and supported by a community of control system developers distributed across several national laboratories. The SNS started operations using EPICS 3.14.7. Various incremental updates have been applied over the years and the current operational version for most IOCs is EPICS 3.14.12.8. EPICS 3.14 is no longer being upgraded.

The EPICS Collaboration worked for nearly a decade to develop EPICS 4, to support a richer set of data types for more advanced devices and applications along with a new communication layer, PVAccess, to support higher bandwidth data transfers. Unfortunately, upgrading from and EPICS 3 system to EPICS 4 is virtually impossible for such a large operational EPICS installation due to the need to upgrade all devices and clients at the same time [5]. Recognizing that the modern features of EPICS 4 would only be available for new installations, the collaboration then produced a combined release, EPICS 7, which allows the simultaneous operation of EPICS 3 and EPICS 4 systems and both corresponding communication protocols within the same EPICS namespace [6]. This gives the ICS a viable upgrade path and work is underway to test EPICS 7 in our operational environment to ensure compatibility prior to widespread IOC and client upgrades.

After the SNS began operations, new operator tools were developed, primarily at the SNS, and shared throughout the EPICS collaboration [7]. These tools, packaged as Control System Studio (CS-Studio), include tools for archiving, data browsing, alarms, display building and graphical user interfaces and feature a common look, feel and behavior along with interoperability which was missing in the earlier tools which were each developed independently. These tools are now used in the SNS Central Control Room, and local control rooms, except for the new display manager for which the migration is in process but requires significant testing time due to the installed base of over 4,500 screens. Many of these screens are used as templates in specific applications or run custom scripts which complicates the conversion process. The new display manager tool is used throughout the SNS instrument beamlines which also have EPICS based control systems.

While software generally requires more frequent update cycles than hardware, this is especially true for graphical tools like operator interfaces. The original CS-Studio toolkit had to be refactored to remove its dependency on the aging Eclipse Rich Client Platform, which resulted in more efficient execution and a better development environment. These tools were adopted for the SNS instrument beamlines when they were upgraded to use EPICS and a new tool, Scan Server, was added for experiment setup and control. In parallel to the CS-Studio desktop version for control room usage, a web display version has been developed. While it is read-only and not meant to support the full feature set of the desktop version, it provides control system and beam line experts with very convenient remote access.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Infrastructure The computing and network infrastructure of the control system benefits from on-going technology refresh. At least 10% and up to 25% of the network switches, servers and workstations are replaced each year to keep the equipment within the expected lifetime limits.

In addition to the in-place upgrade of network equipment, the overall network layout would benefit from a certain level of redesign. The original SNS ICS network was based on several VLANs. Over time, all VLANs have been merged into one large broadcast domain that EPICS servers and clients use to interconnect. The latency of the large domain has reached the point where network clients regularly need to broadcast several search requests before they see a reply. A redesign which for example separates the different types of network traffic and thus isolates the EPICS communication will be necessary before extending the existing network to cover many additional devices.

The servers and workstations are currently undergoing a major revision of the underlying operating system to 64-bit Linux as hardware is no longer available to run the older 32-bit versions. This requires recompiling the substantial installed base of server hosted software and extensive testing with machine operators to ensure functionality is maintained while this upgrade to a supported version of Linux is implemented. The advent of EPICS 7 is another motivation for developing a different network topology to fully benefit from the more efficient higher data transfer rate which should be expected from newer devices.

Along with the Linux upgrade, the software development environment is being restructured and the original CVS code management system is being replaced by Git, a more modern tool with a richer code management feature set. These changes will make software development, which may require multiple branches, easier to manage and merge into a final product for release. It will also simplify the process of installing different software versions.

Slow Controls

Industrial process controls solutions are used on the device control level for the many machine systems which do not require real-time functionality. These include the superconducting cryomodules, cryogenics plant, target facility, vacuum, and conventional facilities. Many of the PLC processors used in these systems were upgraded in 2009 following failures that were tied to manufacturing issues. While these systems have proven reliable, most of our installed base is at or near end-of-life when spares and firmware patches will no longer be available, leaving these systems vulnerable especially to security issues. Additionally, the underlying network technology (ControlNet, DeviceNet) exposes weaknesses that can be rectified by upgrading to the more modern Ethernet solution. In some cases, the segmentation of these systems, either by design or growth, is not ideal. Hosting multiple unrelated applications on a single processor expands the consequences of a single failure therefore these systems will be segregated. The future upgrade path includes new processors and I/O modules to run supportable firmware and reduce the probability of hardware failure during operations.

To minimize disruption, PLCs upgrades will be packaged to install new hardware and replace ControlNet or DeviceNet communication with Ethernet at the same time. Multiple PLCs are typically grouped into ControlNet clusters and all PLCs and communication modules in a cluster will be upgraded at the same time to avoid rescheduling ControlNet multiple times and the extra effort otherwise needed to reprogram and test the PLCs for each change. DeviceNet upgrades may prove more challenging due to the communication module being embedded in vendor supplied equipment. The small number of PLCs which host multiple unrelated applications will be prioritized for upgrades first. Due to resource availability, this work will most like follow the substantial completion of PPU controls in 2023.

Real-time Systems

The control system applications that require performance beyond that possible using industrial process controls technology were originally built using VME hardware and the VxWorks operating system. The original Motorola 2100 model processors lacked the required performance for some applications, most notably the LLRF. This model also suffers from poor network stack performance and is particularly vulnerable to reconnection failures. These processors are being replaced with the 5500 model which successfully addresses both issues. An upgraded VxWorks kernel is available to support the new hardware and fix the network stack issues. About 70% of the original processors have been replaced so far.

Some of these high-performance systems, built on VME or custom technology, have started to experience obsolescence issues and we have developed a common µTCA based approach to upgrade these systems. The concept was initially proven on the Ring LLRF which was originally very different from the Linac LLRF and suffered from early obsolescence and sustainability issues. The new Linac LLRF system being developed for PPU follows the µTCA model along with the new MPS and recent upgrades for the high-performance waveform generation and monitoring systems. Originally developed for the high energy physics community, µTCA provides a commercially available open-standard architecture with built in high availability options such as redundant power supplies and cooling. With a variety of commercially available modules and features to support high speed applications, µTCA is increasingly selected for new accelerator real-time systems providing an opportunity for collaboration. The nine systems installed so far have proven reliable.

Personnel Protection Systems

The SNS Personnel Protection Systems (PPS) are credited engineered controls designed to prevent people from harm due to hazards associated with operating the accelerator such as radiation and oxygen deficient environments. This is primarily achieved by providing access controls and interlocks linked to the machine status and monitoring

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR01

radiation and oxygen levels. In the case of unexpected, elevated radiation levels, the machine is shut down and the source investigated. In the case of depleted oxygen levels, alarming beacons are activated triggering an area evacuation and investigation.

These systems were originally engineered using the same industrial controls technology discussed in the Slow Controls section of this paper but deployed on isolated networks and with additional processes in place for the development and testing of credited systems. These systems are being updated to use more recently available safety rated PLCs and I/O and Ethernet based communication modules. The PPS for at least one machine sector is upgraded each year. Recently, these upgrades have been combined with the requirement to provide additional PPS I/O for the PPU project. The radiation monitors are obsolete and will be replaced with commercial units and a low maintenance ODH sensor design is underway.

CONCLUSION

The three-tier architecture and distributed nature of the SNS EPICS based ICS has proven reliable for the first fifteen years of SNS operations. As the systems age, hardware and software can be upgraded with new designs in a minimally disruptive manner, during scheduled machine outages, by evolving layers and components individually or in groups. EPICS 7 provides a viable upgrade path for the run-time execution engine and communications layer. The configurable nature of the operational tools allows new displays to be created and added during operations and new alarm and archive parameters can be added by inserting an entry in a database even during machine operations. While this architecture is serving the facility well, it is clear many systems require technology updates in the coming years due to obsolescence.

While PLCs and IOCs can be replaced without changes to the underlying system, significant engineering effort in the form of design, documentation and testing is required to provide reliable replacement systems therefore the upgrade efforts will be paced by resource availability and should be considered an ongoing effort. With careful prioritization and adequate resource investment, procurement and labor, the ICS can evolve and serve the SNS well into the future. Confidence in this architecture and technical implementation is evidenced by the selection of EPICS for upgrades to the SNS instrument beamlines and the design of the control system for the STS which leverages the same EPICS based architecture of the original SNS ICS. Key to the success of the SNS ICS has been the free exchange of software and cooperative development efforts which are the hallmark of the EPICS Collaboration.

ACKNOWLEDGEMENTS

Sincere thanks to the countless engineers, designers and technicians who have designed, built, maintained, and upgraded the SNS control system over the last twenty years and to EPICS developers everywhere who not only share their code, but also generously contribute their time and share their experience help others solve problems. The fingerprints on the SNS ICS come not only from SNS staff, but from the partner laboratories and other EPICS sites. The many exceptional individuals working together has truly made an unlikely collaboration work beyond what could have been imagined when EPICS was born over 32 years ago.

REFERENCES

[1] D. P. Gurd, “Management of a Large Distributed Control System Development Project”, in *Proc. 8th Int. Conf. on Accelera-tor and Large Experimental Physics Control Systems (ICALEPCS’01)*, San Jose, CA, Nov 27-30, 2001, paper TUAP062, pp. 58-62.

[2] S. M. Hartman, K. S. White, “Control System Plans for SNS Upgrade Projects”, *in Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’19)*, New York, NY, Oct 5-11, 2019. doi:10.18429/JACoW-ICALEPCS2019-MOAPP03

[3] R. Nelson, B. Oerter, T. Shea and C. Sibley, “SNS Timing System”, in *Proc. 8th Int. Conf. on Accelerator and Large Ex-perimental Physics Control Systems (ICALEPCS’01)*, San Jose, CA, Nov 27-30, 2001, pp. 629-631.

[4] D. Thompson, D. Curry, and J. Dedic, “Timing System Update for SNS”, *in Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’09)*, Kobe, Japan, Oct 12-16, 2009, pp. 483-485.

[5] A.N. Johnson et al., “EPICS 7 Core Status Report,” *in Proc17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’19)*, New York, NY, Oct 5-11, pp. 923-927. doi:10.18429/JACoW-ICALEPCS2019-WECPR01

[6] G. White, et al., "The EPICS Software Framework Moves from Controls to Physics.", *in Proc. 10th International Particle Accelerator Conference (IPAC 2019)*, Melbourne, Australia, Jul 14-17, 2019, pp. 1216-1218. doi:10.18429/JACoW-IPAC2019-TUZZPLM3

[7] K. Kasemir, “Control System Studio Applications”, *in Proc 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’07)*, Knoxville, TN, Oct 12-16, 2007, paper ROPB02, pp. 692-694.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

MODERNIZING DIGITAL VIDEO SYSTEMS AT THE NATIONAL IGNITION FACILITY (NIF): SUCCESS STORIES, OPEN CHALLENGES AND FUTURE DIRECTIONS

V. Gopalan, A. Barnes, G. Brunton, J. Dixon, C. M. Estes, M. Fedorov,
M. Flegel, B. Hackel, D. Koning, S. Townsend, D. Tucker, J. Vaher,
Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550, USA

Abstract

The National Ignition Facility (NIF), the world's most energetic laser, completed a multi-year project for migrating control software platforms from Ada to Java in 2019. Following that work, a technology refresh of NIF's Digital Video (DVID) systems was identified as the next important step. The DVIDs were facing long-term maintenance risk due to its obsolete Window XP platform, with over 500 computers to be individually upgraded and patched, 24 camera types with a variety of I/O interfaces and proprietary drivers/software with their licensing needs. In this presentation, we discuss how we leveraged the strengths of NIF's distributed, cross platform architecture and our system migration expertise to migrate the DVID platforms to diskless clients booting off a single purpose-built immutable Linux image and replacing proprietary camera drivers with open-source drivers. The in-place upgrades with well-defined fallback strategies ensured minimal impact to the continuous 24/7 shot operations. We will also present our strategy for continuous build, test, and release of the Linux OS image to keep up with future security patches and package upgrades.

INTRODUCTION

Digital Video (DVID) systems are an integral part of the NIF control system. They participate in a variety of automatic loops (e.g., automatic alignment), provide critical diagnostics to study the laser, and allow operators to observe the state of the system in real time. The cameras may be used for still captures, streaming video or both. The capture may be precisely timed using triggers, synchronized to the arrival of the laser pulse, or may be manually captured by an operator on demand. It can be used for spatial measurements – e.g., for measuring the spatial aberrations of the NIF beam's optical wavefront. With over 500 DVIDs deployed along the laser path leading to the target, the DVIDs provide machine vision functions to NIF including optics inspection, automation, application-oriented machine vision processing, and vision-guided automatic alignment/positioning systems.

LONG TERM MAINTENANCE CHALLENGE

The NIF Integrated Computer Control System (ICCS) underwent a phase of modernization which concluded in 2019 [1]. As part of that work, the low-level, hardware-facing Front-End-Processors (FEPs) were migrated from Ada to Java based software and associated Java frameworks and libraries. The DVID FEPs also benefitted from the overall NIF ICCS modernization, however, the unique nature of the DVIDs called for further upgrades to DVID hardware and software to counter obsolescence and avoid future maintenance jeopardy. The various challenges specific to DVIDs are described in the following sections.

Obsolete OS

The DVID FEPs run on the unsupported Windows XP Operating System (OS) with inherent security issues and lack of hardware support as newer computer hardware discontinue support for older operating system versions. Non-availability of machines that can install Windows XP on them means that replacing or upgrading hardware will only become a bigger challenge with time. Stop gap measures such as upgrading random access memory (RAM) in existing systems to improve application performance also does not help much due to XP's limitation on maximum supported memory (4GB).

Large Number of FEP Images

Each DVID FEP has its own Windows image installed in the local hard disk and these OS images are required to be individually upgraded and patched. Deployment of OS/driver upgrades and patches are expensive and difficult due to the large number of machines that needs to be individually patched and tested for every modification.

Multiple Camera Interface Types

The DVIDs use several interface types to talk to the different types of cameras. Figure 1 shows how the interface types are split across all the DVID FEPs.

Interface type	No. of FEPs
FireWire	487
GigE	318
Analog-FireWire	110
Analog-PCIe	2

Figure 1: Camera interface types and usage count.

Cameras that use FireWire, also commonly known as IEEE 1394, use DCAM protocol that describes the exchange of data over the FireWire bus interface. GigE based cameras use GigE Vision [2], a standard for video transfer and device control over Ethernet networks. Analog cameras typically provide an RS-170 analog signal, which then

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR02

gets digitized for further image processing by using a Video-To-FireWire converter or a PCI Express (PCIe) frame grabber computer adapter card.

Multiple Camera Types

The NIF control system employs multiple camera types from multiple manufacturers as shown in Fig. 2. The selection of a particular camera is based on the requirements for the specific application associated with the control function such as image resolution, streaming frame rates, or specialized features such as radiation or vibration tolerance. In some specific use cases, cameras have NIF specific customizations to satisfy the requirements for use in the control system. Each of these camera types comes with its associated Windows drivers/software and often have proprietary licensing schemes that are different across the different manufacturers or camera software vendor.

Interface type	Camera Manufacturers	Camera Models
FireWire	4	7
GigE	4	15
Analog-FireWire	1	1
Analog-PCIe	1	1

Figure 2: NIF DVID camera types.

Hard Disks – A Potential Point of Failure

The hard disks in the DVIDs are a potential point of failure and can cause downtimes due to data loss or corruption. With over 500 DVIDs deployed, the collective MTBF of the disks is a potential risk to the NIF's high expectations for availability and reliability, which essentially dictates a "zero tolerance for error". Disk based systems continue to pose a challenge to meet this level of fault tolerance, unless fault tolerant, redundant storage is implemented in every DVID.

Aging FEP Computer Hardware

The deployed FEP machines belong to different generations of CPU and hardware, and many of them do not have enough memory (RAM) or CPU horsepower to run a modern OS. Regardless of the OS upgrade path chosen, newer OS technologies demand more capable machines.

From the perspective of running a modern OS on it, Fig. 3 shows the percentage of FEP hardware that needs to be replaced or upgraded.

POTENTIAL STRATEGIES, OPTIONS AND CONTRAINTS

Hardware/software changes typically require facility downtime to support modifications, tests, and qualification. Continuous run of experiments at NIF means that an extended downtime for upgrade of hardware and software of the video systems is not an acceptable option. The DVID

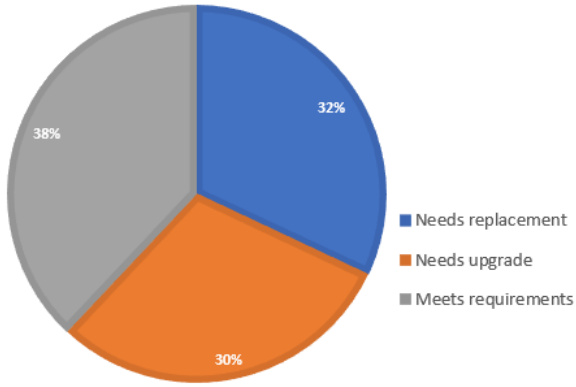


Figure 3: DVID FEP machines.

upgrades will also need to be "In-Place" in parallel with scientific operations – an approach that the facility has tolerated well as part of the prior NIF modernization phase [1].

With the success of "In-Place" upgrade strategy as described in [1], the digital video control system modernization naturally adopts the same overall strategy and best practices, with incremental deployment of new hardware/software components keeping compatibility with the existing system. This allows a smooth upgrade and progressive replacement with minimal impact to the running system.

OS Upgrade / Replacement

With the DVIDs currently running Windows XP, the obvious upgrade path may appear to be to install a newer version of Windows. However, this will not solve the challenge of maintaining a large number of separate OS images. Furthermore, opensource support for Windows based driver software for cameras is limited, and the proprietary nature of Windows will limit our ability to create a custom OS image with tightly controlled image size, tailored to run on the specific set of FEP hardware with available memory and CPU performance constraints.

A compelling option for DVID OS candidate originated from an ICCS maintained custom Linux OS distribution based on Gentoo Linux [3] that has already been successfully deployed to VME FEPs in NIF, including support for network boot mode. The already available Diskless Gentoo OS (DGOS1) offered us a platform where we would be able to incrementally add support for specific features required by DVID systems.

The 3rd option was to explore other Linux distributions such as RHEL, Debian and other Linux distributions. With Gentoo based systems already successfully deployed in NIF, this alternative option which involves significant additional effort, would be justified only if the existing Gentoo based OS could not be enhanced to meet the DVID requirements.

Even if Linux is selected as the primary DVID OS, the possibility of having specialized DVID systems that can only function with proprietary/commercial software needs to be considered as well. We expect a small percentage of

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

DVIDs (less than 5%) that may still need a Windows based solution to support specialized camera, drivers, or image processing software.

Camera Drivers

There were two broad categories to consider: Open-source drivers and proprietary drivers.

Proprietary Drivers Proprietary drivers include drivers provided by the camera manufacturer or supplied by a third-party software vendor. The proprietary drivers typically come with license costs and restrictions (e.g., node locked, hardware dongles, license built into camera).

Open-Source Drivers This includes drivers available with one of the many open-source licenses either distributed via open internet (e.g., through GitHub), or distributed by the manufacturer itself in source code.

The manufacturer provided drivers only work with the specific vendor's cameras, so even though the driver is standard compliant, it is generally not possible to use it across cameras from different vendors.

With the above considerations, the best approach was determined to be using open-source drivers for FireWire and GigE cameras, that would be manufacturer / model agnostic, if the cameras are standards compliant.

Camera Maintenance Tools

To enable testing of cameras in production during installation, repair or troubleshooting, operators need GUI (Graphical User Interface) tools that can query the camera parameters, capture image, and perform streaming tests. The Windows DVID with the manufacturer or third-party vendor supplied software provides a variety of GUI tools for debugging the cameras outside the ICCS framework. If we replace Windows with Linux and open-source drivers, these tools must be replaced with equivalent functionality. Since the Diskless DVID will be headless, it is also required to launch the GUI remotely.

We considered the following options for developing the camera GUI tools:

1. Use GUI sample applications that come with the opensource camera drivers. The UI must be launched remotely using X11 forwarding.
2. Use manufacturer provided Linux based applications, either by running binaries or building from source. The UI must be launched remotely using X11 forwarding.
3. Build our own custom GUI app based on web-based technologies and launch the UI remotely on the client web browser.

The UI application in option 1 and 2 above has dependencies on additional Linux libraries such as Gnome, SDL and others, and the Gentoo system should support these dependencies so that the app can be launched successfully. This could increase the OS image size significantly.

MODERNIZED ARCHITECTURE

FEP Operating System

The legacy Windows OS will be replaced with Linux OS, which will be a single image that will be configured and built to run on all DVIDs. This new Diskless Gentoo OS Version 2 (DGOS2) will be derived from the VME diskless Gentoo (DGOS1) with the following enhancements:

- Upgrade to latest Gentoo packages
- The CPU architecture type is selected from an Intel architecture version that ensures compatibility with existing DVID machines. The "core2" architecture was chosen, which provides a good balance of compatibility with older processors while not taking a performance hit on the newer processors with architecture later than core2.
- Upgrade Linux kernel to a long-term support (LTS) version: v4.14 LTS with Projected EOL Jan 2024 was chosen.
- Enable FireWire support in the kernel.
- Support additional network interfaces required for GigE DVIDs. This requires a private subnet to be created for attaching GigE cameras.
- Tune kernel network buffer size to optimize GigE performance and prevent frame drops.
- Enable NIS user authentication.
- Add FireWire hot-plug support.

The distribution builder builds all the packages from source with the selected CPU architecture. This is required, since a package binary that is built for a slightly different hardware configuration can cause the boot to fail or specific packages to malfunction.

Booting Mechanism

The DVIDs boot off a common Linux OS image over the network. The network boot is implemented using Preboot Execution Environment (PXE) [4], Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP).

When the diskless DVID client is booted up, the PXE boot process will request the IP Address from the DHCP Server (Fig. 4). The DHCP server responds by sending the

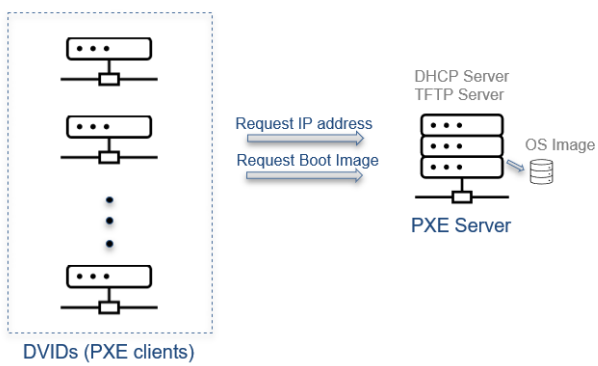


Figure 4: Diskless clients boot architecture.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR02

IP Address and information about the boot image. In the second step of PXE boot, the boot image is requested and loaded into the DVID's Random Access Memory (RAM) by using TFTP. After all the image data loaded, the DVID automatically loads the OS and executes the OS boot sequence.

Driver Level Redesign

The proprietary drivers in the FEP will be swapped out and replaced with opensource variants. After prototyping and verifying functionality and performance, the following opensource drivers were selected:

- FireWire: **libdc1394** library with support of over 500 camera models, provides a complete API for controlling cameras conforming to DCAM specifications.
- GigE: **Aravis** library implements the Gigabit Ethernet protocols used by GigE cameras.

Although the opensource drivers support most of the required features, there are specific instances where we had to enhance it. For e.g., libdc1394 does not support access to shutter time base register which is required for supporting full range of exposure times for some cameras.

CORBA Interfaces and FEP Java Code

ICCS uses CORBA middleware technology to build a distributed control system network. Carrying over CORBA IDL interfaces intact from legacy Windows to Linux FEP software is a vital requirement for achieving smooth In-Place migration [1].

Existing Windows based FEPs were designed such that the Java support classes were selected and instantiated based on the camera vendor. Each support class is an API class whose native methods were Java Native Interface (JNI) calls to a lightweight ICCS-maintained wrapper that dispatches the calls to vendor-supplied and vendor-maintained library calls (DLL) for command/control.

Moving to Linux requires a re-architecture since most camera vendors do not provide out-of-the-box support for Linux. ICCS already maintains a vendor-agnostic approach to instantiating support classes. We designed the Linux FEP to leverage the existing methodology, abstract it to support all new Linux DVID FEPs including generic support for multiple camera bus types.

Cameras are described by text-based, human-readable YAML files, with one YAML file per camera make/model which is then stored in the file system. These camera description files are controlled alongside source code, deployed to production along with software release and gets loaded and parsed by FEP during initialization. Contents of the file determines support class instantiation and supported feature configuration for command and control of each camera defined per process in the database. FEP loads ICCS-maintained thin wrapper library (written in C, compiled for Linux) that dispatches native calls from Java FEP to the FireWire/GigE driver.

FEP Hardware Upgrade

For each DVID that is converted to Diskless Linux, the FEP computer will be either replaced with a newer machine, upgraded with higher RAM (16 GB) or used without any changes. The modification depends on the category of the machine as described in Fig. 3.

Performance Considerations

Two major areas required performance considerations: CPU performance and network performance.

CPU Performance Windows video FEPs used Intel's Integrated Performance Primitives (IPP) library to perform image compression (JPEG) and decimation. Current Windows FEP implementation utilizes IPP version 4 that dates back to 2003. To be able to run IPP on Linux, we discovered that this requires an update to bring the latest IPP support into the image processing library compiled for Linux. After the update was successfully integrated into the Linux FEP, verification of new image processing libraries was performed with the help of Automatic Alignment tests by comparing centroid locations obtained from test images decimated with current Windows 32-bit version of image processing libraries with those using new 64-bit Linux version of the library.

Image capture and streaming tests (Fig. 5) were conducted with various FEPs to measure, improve and verify that the capture response times, and streaming frame rates met or exceeded the Windows FEPs that were being replaced.

Network Performance The Windows FEP used a vendor supplied performance driver that optimized GigE Vision network traffic for adapters that use specific Intel chipsets. The advantage of the performance driver is that it offloads network traffic processing to the adapter and lowers the CPU load. With Linux FEPs, our testing did not indicate a need for the performance drivers, and the streaming performance was comparable to Windows FEPs without using the performance driver.

host	configuration			fps	measured* fps	diff from windows
	ROI	compression	bit depth			
PixeLINK 959G on Linux FEP, Dell R200)	1600 x 1200	ON	16	4	4	0.14
				5	5	0.35
				6	6	1.04
				7	6.93	1.07
				8	6.92	0.61
				9	6.833	0.433
	1600 x 1200	ON	8	10	8.8	-0.77
				11	8.75	-1.49
				12	8.58	-1.54
				13	8.6	-1.68
				14	8.55	-2.28
	1600 x 1200	OFF	16	4	3.95	0.92
				5	4.77	1.89
				6	5.22	2.32
				7	5.4	2.47
				8	7.54	4.68
				9	7.61	4.78
	1600 x 1200	OFF	8	10	9.46	3.63
				11	10.24	4.49
				12	10.78	4.99
				13	11.55	5.82
				14	13.16	7.33

Figure 5: Fragment of streaming performance test results.

Database and FEP Identity

A change in database schema not required for the Linux conversions since the strategy is to maintain the configuration and capability of the FEP. Reusing the same tables between legacy Windows and new Linux implementations also allows for easy switching between Windows/Linux software.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

However, there will be minor changes to database to configure Linux specific start-up parameters. This will impact the data stored in the tables – but not the schema itself.

Taxonomical name and identity of the FEP will not change and the Linux FEPs will continue to be identified using the same identify of the Windows FEPs that they are replacing. This ensures that the conversion is transparent to the other components in the control system. This aspect is also very important for the In-Place upgrade strategy to work.

VERIFICATION AND DEPLOYMENT

Test and verification are first done in the offline (non-production) environments which consists of development, integration, and QA. One of more setups of each DVID type is available across these environments, which assure robustness of these conversions before we apply it in production. A full suite of manual and automated endurance tests is run and compared against the corresponding Windows FEP to ensure that the In-Place replacement will work as expected.

Deployment to NIF requires extensive coordination between controls hardware teams, IT, operators, subsystem leads, and testers during conversion. A typical Windows DVID FEP to Linux conversion consists of the following steps:

1. Shut down FEP's running processes
2. Configure BIOS to PXE boot instead of booting from disk.
3. Configure DHCP server to add the FEP's IP and PXE boot configuration
4. Reboot the FEP and verify successful boot-up into Diskless Linux.
5. Apply database changes to modify Linux FEP configuration parameters.
6. Startup the FEP's processes
7. Verify functionality using image capture and streaming tests.

Status of Deployments to NIF

The Diskless Linux solution has been successfully deployed to DVIDs in NIF, and functionality and performance is now proven by operation in the NIF's control system that is running a 24x7 scientific experimental schedule. With the modernized solution, our team has successfully accomplished a technology refresh of NIF's Digital Video systems and effectively solved the Digital Video systems' long term maintenance challenge.

FUTURE DIRECTIONS

Diskless Gentoo OS Roadmap

The official Gentoo Linux releases use a rolling release model, where packages are made available to the distribution as soon as they are released upstream. It has been a challenge to keep DGOS up to date with the official releases, since new versions of DGOS are currently released only as needed - typically for adding specific new features required by NIF or for applying security patches. Going forward, our plan is to release DGOS at least every 6 months to keep it current with the official Gentoo packages.

The DGOS releases are currently not integrated into the NIF ICCS continuous integration (CI) system, and verification depends primarily on manual tests. CI integration is a work in progress, and we plan to progressively integrate the DGOS build, test and release into the overall CI system which will enable us to support faster rate of release and quicker fault isolation.

Diskless Video FEP Enhancements

As part of the diskless video conversions, a large subset of DVIDs were successfully configured to handle two cameras simultaneously, significantly reducing the number of deployed FEP hardware which reduces maintenance costs and thermal stress. The upgraded DVID computers offer the possibility of further consolidation of camera functionality into the same FEP (for e.g., handle 4 cameras in the same DVID). The possibility of such efficiency improvements will be explored in the future.

Current implementation only supports FireWire and Gig cameras. Support for analog/PCIe cameras is planned in the future.

Camera Maintenance GUI Tools

We are working on deploying a maintenance GUI framework, based on web technologies, that will securely expose the camera command and control over a RESTful API, and allow testing of cameras in production during installation, repair or troubleshooting remotely from a web browser.

ACKNOWLEDGMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-827589.

REFERENCES

[1] M. Fedorov et al., "In-Place Technology Replacement of a 24x7 Operational Facility: Key Lessons Learned and Success Strategies From the NIF Control System Modernization", *presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Oct. 2019, paper WEDPL01.

[2] GigE Vision Standard, https://www.automate.org/a3-con-tent/vision-standards-gige-vision

[3] Gentoo Linux, https://www.gentoo.org/

[4] Preboot Execution Environment (PXE) Specification, http://www.pix.net/software/pxeboot/archive/pxespec.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

LOFAR2.0: STATION CONTROL UPGRADE

T. Juerges[1,2*], J. D. Mol[2†], T. Snijder[2‡]
[1]Square Kilometre Array Observatory (SKAO),
Jodrell Bank, Macclesfield, SK11 9FT, United Kingdom
[2]Netherlands Institute for Radio Astronomy (ASTRON),
Oude Hoogeveensedijk 4, 7991 PD Dwingeloo, Netherlands

Abstract

After 10 years of operation, the LOw Frequency ARray (LOFAR) telescope is undergoing a significant hardware upgrade towards LOFAR2.0. The hardware upgrade will enable the phased array telescope to observe at 10-90 MHz and at 120-240 MHz frequencies at the same time. With the upgrade comes also the chance to review LOFAR's Control System and to make it ready for the next 10 years of operation at the forefront of low-frequency astronomy. In this work we will give a brief overview over the LOFAR telescope with its more than 50 geographically distributed receiver locations (LOFAR Stations), and the software that is necessary to monitor and control every single one of them. We will then describe the Station Control architecture, with its software design and how it is implemented in Python 3 with Tango Controls, OPC-UA clients and deployed as Docker containers. Lastly we will report on the successful use of open stack software like ELK and, Grafana.

LOFAR TELESCOPE OVERVIEW

LOFAR [1] is a geographically distributed radio telescope array, consisting of around 60,000 dipole antennas. The antennas are grouped into 56 *stations*, 38 of which are deployed in the Netherlands, and the remaining 14 in other countries across Europe. The scientific data from these stations are streamed to our real-time GPU correlator [2] in Groningen, the Netherlands. Thusly correlated (and beamformed) data products are subsequently post processed. We send the end result to our tape archives in the Netherlands, Germany, and Poland. There the data products are made available for download by the scientists.

LOFAR2.0 Station Upgrade

A LOFAR2.0 station will, like the current LOFAR stations, consist of up to 96 high-band dipole tiles (110–250 MHz), and 96 low-band dipole antennas (10–80 MHz). The tiles and antennas are connected to 64 Receiver Control Units (RCUs), which apply a configurable analog filter.

LOFAR2.0 will redesign these RCUs to have improved filters. The new RCUs will also have the ability to process data from all antennas simultaneously [3].

The RCU output is sent to station signal-processing boards, to be beamformed and converted into UDP packets. These packets are streamed over 10 GbE fibres to the correlator.

Station Signal Processing

A station will contain up to 8 Uniboard^2 processing boards [4]. The boards use 32 FPGAs in total to sample and digitise the signal at 200 MHz and calibrate, exchange, beam form, and correlate their input. The end result is a 3–9 GBit/s data stream to the correlator per station, as well as up to 300 Mbit/s of statistical information.

STATION MONITORING AND CONTROL

The hardware in each station exposes tens of thousands of monitoring and control points through various interfaces and protocols. Basically the Monitoring and Control of a LOFAR2.0 station can be condensed into two simple operations at a station:

- Modify the behaviour of our hardware over time, e.g. point at different sources in the sky.
- Verify that the dynamic behaviour has been successfully modified.

In addition to the basic concepts of station operation, the nature of the distributed telescope requires that we also keep track of the system health and let the station autonomously act on extreme scenarios such as overheating of the equipment.

Finally, we are interested in monitoring the quality of the data recorded through our antennas and produced by our processing boards. To this purpose, the signal-processing boards continuously emit statistical information from several points in the signal chain.

OPC UA as a Common Hardware Interface

The hardware that is to be monitored and controlled in a LOFAR2.0 station comes in various shapes and forms. This could imply that a station's Monitor and Control system would have to support a variety of different hardware interfaces and protocols. We have, for example:

- Uniboard^2 processing boards: I^{2}C
- FPGAs on Uniboard^2: UCP (Uniboard Control Protocol) over IP
- RCUs: I^{2}C
- Power supplies: PLC interface
- Temperature sensors: PLC interface
- Network switch: SNMP

From prior experience in LOFAR1, as well as in other telescope monitor and control systems (ALMA, WSRT), we

* thomas.juerges@skao.int
† mol@astron.nl
‡ snijder@astron.nl

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

learned that a common hardware communications protocol minimises software complexity. Engineering and maintenance personnel also benefits from a common hardware interface on site as well as remote. The tooling can be unified and be kept simple to use, yet powerful enough to debug hardware at an acceptable low enough level.

Thus we expect a common hardware protocol to have the following properties:

- Open tooling available for remote hardware and interface debugging
- No need for specialised equipment to communicate with the hardware (JTAG dongles, I^2C cards, etc.)
- Protocol client implementation available in Python 3 to integrate seamlessly with the Station Control software
- Single server supports multiple clients
- Support for a direct mapping of Monitor and Control points from hardware to software.

The widely adopted OPC UA protocol [5] supports all of the properties above. In addition, OPC UA supports the following aspects in a client-server system:

- OPC UA protocol is based on TCP/IP: Connection reliability, simplicity of software integration
- Protocol insensitive to specific hardware timing: We chose to implement timing sensitive behaviour server side.
- Ability to browse self-describing attributes and methods: Enables introspection of the available Monitor and Control points (name, type, dimensionality)
- Server implementations in C/C++ and Python 3: Allows hardware engineers to implement the servers observing timing specific behaviour and other details of the hardware.

OPC UA is a feature-rich protocol, but does not require the user to make use of the full set of features. For LOFAR2.0 we chose to limit the use of OPC UA features to:

- Browsing of attributes and methods
- Reading and writing of attributes
- Calling of methods
- Use of OPC UA native data types only

OPC UA Hardware Translators

We run a dozen of OPC UA servers on Raspberry Pis in the LOFAR2.0 station as interfaces to the station's hardware. These servers „translate" the hardware-specific communications protocols[1] into OPC UA attributes and methods. Thus the name Hardware Translator, or Translator for short, for the OPC UA server.

HIGHER-LEVEL MONITOR AND CONTROL WITH TANGO CONTROLS

For the LOFAR2.0 Station Control software Tango Controls [6], an open source and distributed object-oriented Monitor and Control system, has been chosen as the core software framework. It enables the software engineers to focus on designing and implementing feature rich Monitor and Control application hierarchies, that consist of an arbitrary number of abstraction levels. The software engineer does not have to implement low level functionality, like for example automatic polling of an attribute's value from a source, attribute value on-change events, attribute value alarm subscriptions or a system wide logging system. It also provides a transparent and configuration-independent component discovery through its underlying CORBA [7] foundation. Tango Controls also provides a lot of tooling for rapid prototyping and supports native Python 3 bindings.

Devices and Device Servers

A LOFAR2.0 station has no moving parts — the station is controlled through Station Control, the software stack hosted on a computer located in each station. It manages the thousands of control and monitoring points exposed by the Hardware Translators.

In order to map the Translators into the Tango Controls realm, we represent each Translator in Tango Controls as one or more Devices. These Devices are implemented in Python 3 with PyTango [8], the high-level Python 3 API for Tango Controls.

Devices are executed in Device Servers, which can run one or more Devices using multithreading. In order to avoid a shared process space and thus shared crashes and performance bottlenecks, we chose to run each Device in a dedicated Device Server.

The Station Control Devices

We designed a hierarchy of Devices that together form the Station Control Devices landscape, as shown in Fig. 1. The lowest layer Devices communicate with the Hardware Translators, while the highest layer exposes business-logic functionality to our central Telescope Manager.

Especially our signal-processing boards expose a lot of functionality through numeric properties that can be set. In order to keep the responsibilities of each Device manageable, we divide the functionality exposed by some of the Hardware Translators into multiple Devices. OPC UA makes this a seamless experience because it does not enforce and one-to-one mapping between clients and servers.

To further limit the number of attributes per Device, we group most hardware Monitor points into arrays. For example, in one Translator, an array of 16 booleans can flag communication issues with the 16 FPGAs that the Translator monitors and controls. About a hundred arrays remain, ranging from tens to tens of thousands of elements per array.

Tango Controls Attribute Wrapper

The multitude of Tango Controls attributes forced us to look at representing them as efficiently as possible in our source code. For that we developed a generic „attribute wrapper" framework. It allows us to map almost any source of information to a set of Tango Controls attributes. The Attribute Wrapper takes care of the interaction and management of the data source, while the software engineer provides

[1] See section "OPC UA as a common hardware interface".

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

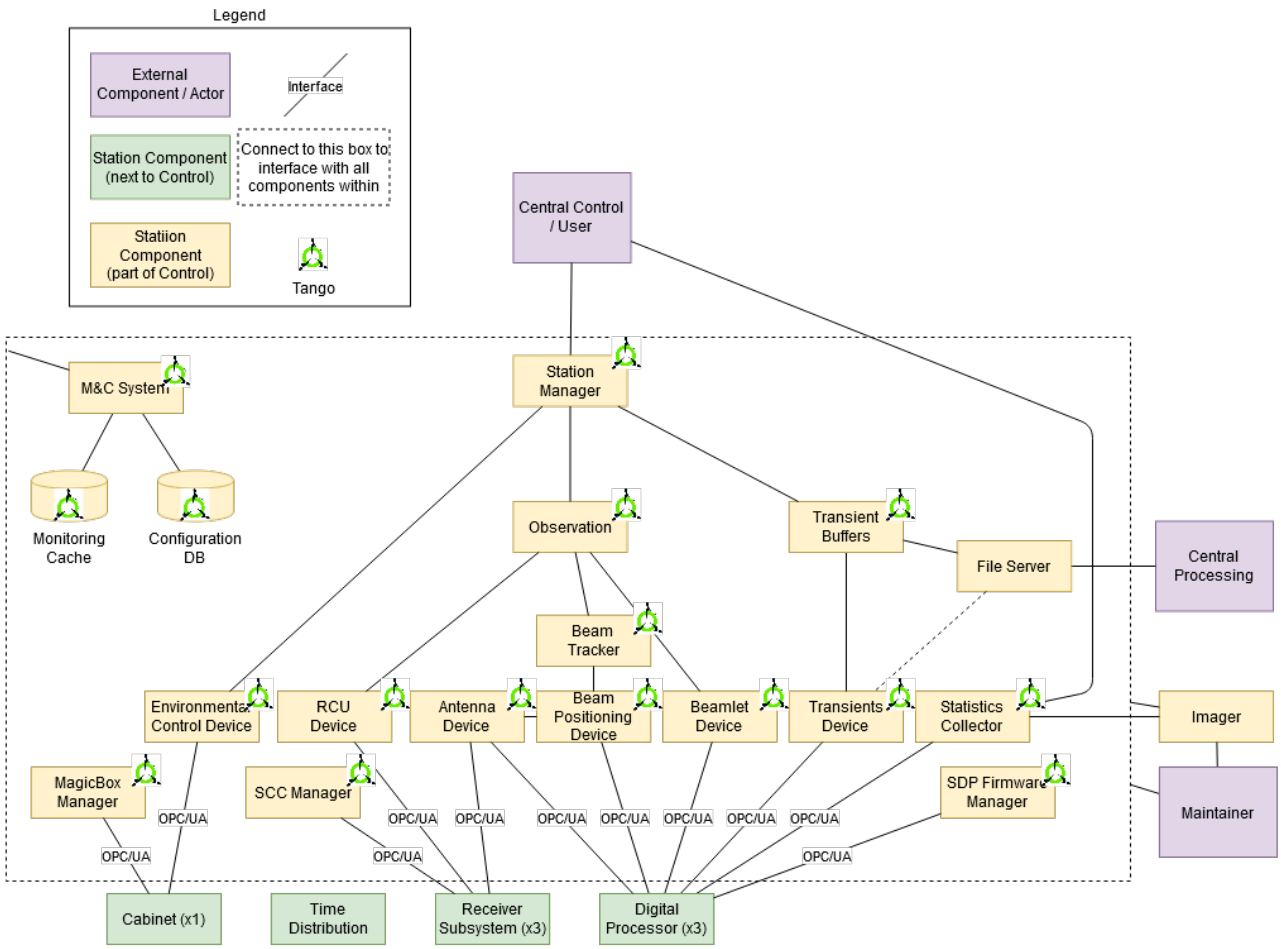


Figure 1: Station Control Devices landscape. The yellow boxes represent software components. Components belonging to Tango Controls, as well as our Tango Devices, are annotated with the Tango logo. The green boxes represent collections of Hardware Translators. Purple boxes represent LOFAR2.0 components external to the station.

just a minimum amount of information when constructing an attribute. The example source code in Fig. 2 shows just two lines of source code that fully constructs two complex Tango Controls attributes. One attribute just reads its value from an OPC UA server, the other one reads from and writes to an OPC UA server.

The Attribute Wrapper requires a protocol or interface-specific implementation. This is as simple as implementing the following functions once per type of interface:

- Connect to the data source
- Disconnect from the data source
- Generic read function that reads a value from a location of the data source
- Generic write function that writes a value to a location of the data source

As seen, in the case of an OPC UA data source for attributes, the attribute construction with the Attribute Wrapper requires just a simple string annotation that tells the built-into the Attribute Wrapper OPC UA client where it can find the respective OPC UA attribute in the server. Once the OPC UA connection is established in the device's initialisation phase, the attributes are automatically linked to

```
RCU_temperature_R = attribute_wrapper(
    comms_annotation = ["RECV", "RCU_temperature_R"],
    datatype = numpy.float64, dims = (32,))
HBA_element_beamformer_delays_RW = attribute_wrapper(
    comms_annotation = ["RECV", "HBA_element_beamformer_delays_RW"],
    datatype = numpy.int64, dims = (32, 96), access = AttrWriteType.READ_WRITE)
```

Figure 2: Example code showing the definition of two attributes in a Tango Controls Device. The `comms_annotation` field encodes the OPC UA specific parameters the OPC UA connection class needs to access this attribute.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the actual OPC UA connection and the relevant OPC UA attribute (and unlinked on de-initialisation).

The result is a very efficient and simple to use framework which we extended to cover the management of, among others, SNMP devices, Docker containers and INI-format files.

Statistical Data Streams

The FPGAs produce streams of statistical data about their main 3–9 Gbps data streams flowing out of the station: crosslet statistics (XSTs) revealing phase and amplitude differences between the inputs, frequency & amplitude plots of each individual antenna (SSTs), and power spectra of the output data stream (BSTs). These statistical data are divided over UDP packets, sent collectively by all FPGAs.

The FPGAs require to be configured with the MAC and IP address of the receiver, of which there can be only one, as UDP Multicast is tricky to set up, especially across switch boundaries. Yet multiple users are interested in these statistics: telescope operators and maintainers use them to verify the data quality and expert users record the stream to produce additional scientific data products.

In LOFAR1, we allow the UDP stream to be configured by any interested user, but learned that such a solution gives an awful user experience. Receiving and recording UDP streams tends to require low-level programming, easily leads to data loss and demands that the user does not forget to turn off the stream explicitly.

The conclusion from past experiences is that we designed a more elegant solution. Each type of statistical data is configured to be sent to a dedicated Device Server. The Statistics Device, that runs in that Device Server, collects the UDP streams from all FPGAs. With the help of our Attribute Wrapper framework, we easily expose the resulting statistical data matrices, the metadata, such as timestamps and other properties, and meta-statistics, such as packet and error counters.

These statistical data matrices represent the most recent statistical values as computed by the processing board. The Statistics Devices expose the matrices as arrays of f.e. 192x512 floats, or 192x192x8 complex values, which typically get updated every second or faster.

Statistics Replicator Monitoring these matrices automatically is rather computing expensive, so we added another interface to record these statistics over time: We expose a TCP port per type of statistic. Any connecting user receives the raw UDP packet stream over this TCP connection until the user disconnects or the Device is shut down. The Device allows as many connections as the available bandwidth allows. Monitoring points for this „Statistics Replicator“ are exposed through our Attribute Wrapper as well.

Statistics Writer Furthermore, we provide the user with a Statistics Writer. It reuses the same classes to convert the Statistical Data Stream into matrices on the client side. Then it just writes them as an HDF5-file to disk. The HDF5-file contains proper data structures and attributes of the Statistical Data Stream.

Observation and Tracking Devices

A station's main task is to partake in observations performed by one to all stations in concert. Observations carry many configuration settings and require parts of the hardware to be reconfigured every second. The adjustments of hardware parameters are necessary to track the observed source across the sky as Earth rotates. We are in the process of managing this behaviour by implementing a Tracking Device to perform the necessary computations.

For each observation, dedicated Observation and Tracking Devices are dynamically created and started. Both work together to maintain the station in a state such that it fulfills the observational requirements throughout an observation. This is done without interfering with observations that are executed in parallel on the same station. The Tango Controls system seamlessly allows this dynamism.

DEPLOYMENT

The core of our Station Control software stack is based on the Docker image framework for Tango Controls, originally provided by the Square Kilometre Array Observatory (SKAO) [9]. A part of the SKAO framework provides a basic setup of Docker containers that run the core Tango processes and an example Device Server. We have significantly extended this part of the framework, for example, by:

- Making the Device Servers that run inside Docker externally reachable, by adjusting the underlying CORBA [7] parameters
- Running each Device Server in a dedicated Docker container
- Added a rich set of integrations with modern tool chains, such as ELK [10], Prometheus [11], Grafana [12], and Jupyter Notebooks [13] (see below)
- Added an integration and unit-test framework, run from Gitlab CI/CD

Stacking Open Interfaces

Modern toolchains make it easier to attain high levels of integration between complex software stacks. We added a significant number of web interfaces by adding off-the-shelf Docker images that needed only little configuration. The separation of services into Docker containers and linking them through open interfaces proved to be a powerful combination.

Jupyter Notebooks A Jupyter Notebook server [13] allows users to access all control and monitoring points in a station. It comes with PyTango [8] pre-installed and pre-configured for easy access to all Station Control Devices. We provide example Notebooks (see Fig. 3) and the user can save and load their own. Plotting libraries are pre-installed as well which makes the Jupyter Notebooks a powerful engineering interface.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

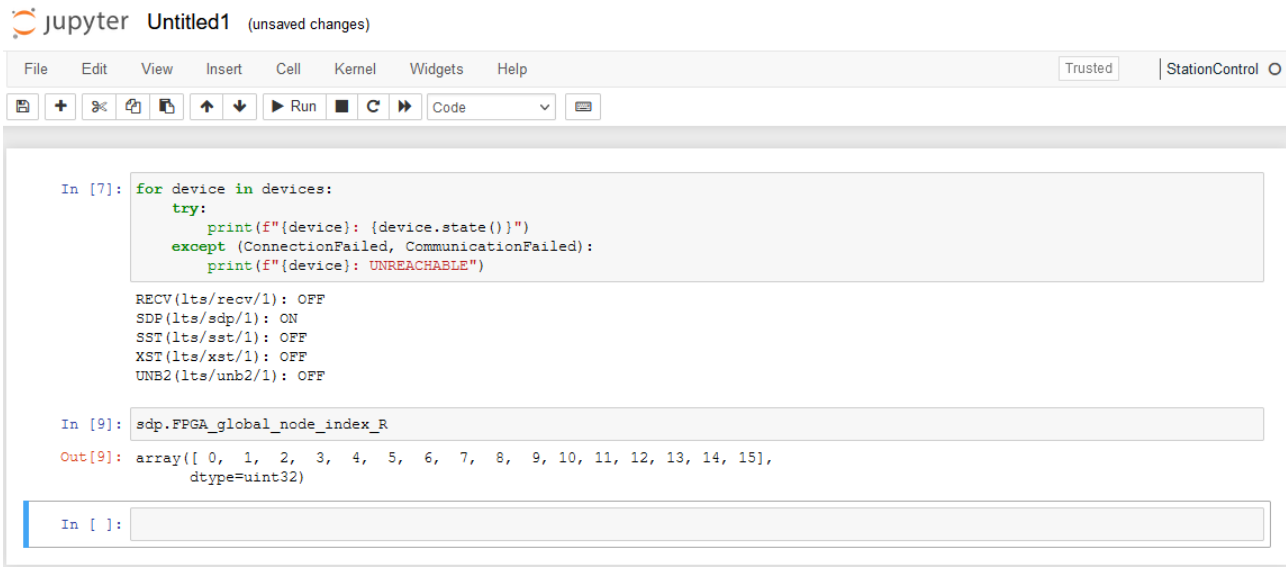


Figure 3: Example of using the station's Jupyter Notebook to read the state of the Devices and one of the Device attributes.

ELK stack An ELK stack [10] collects logs from all Devices, Device Servers and several of the other services that we run. We added Python LogHandlers and Filters to automatically route all Python log output to Tango's log system, our ELK stack and stdout. Each log line is annotated with the name of the Tango Device that generated it. This both leads to richer logs and alleviates the need of having to forward the Tango log streams to our utility classes, as all classes can now just use Python's native logging interface yet still log to Tango.

Grafana We added and adjusted the SKA's experimental TANGO-Grafana exporter [14] to allow all but the biggest arrays of monitoring points to be periodically scraped by a Prometheus time-series server [11]. A Grafana [12] dashboard system (see Fig. 4), also installed on the station, has this Prometheus server configured as its data source, along with the Tango Archiver database, Tango database, and our station's ELK stack. Pre-configured dashboards provide a rich overview of the station's state and state history, from temperature sensors to current hardware settings and active software and firmware versions and the state of all Docker containers.

Sphinx & Read the Docs Finally, we generate user documentation through a Sphinx [15] integration and a web hook from our Gitlab server to Read the Docs [16]. Having our source code publicly available made this trivial to setup.

Figure 4: An excerpt from one of our Grafana dashboards.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Benefits of Simulating the Hardware Interfaces

The LOFAR2.0 station contains custom hardware that the software engineers do not necessarily have access to. We discovered early the benefits of simulation of the interfaces that hardware exposes. Our Translator setup is such, that it allows us to run them stand-alone in Docker containers and simulate the hardware in a very basic and static fashion. While these instances do not have access to backing hardware, they do expose the OPC UA interface just like the actual equipment would. This allows us to write integration tests to verify the attribute names, types, dimensionalities, and other basic properties.

CONCLUSION

Over the past 18 months we found Tango Controls [6] to be a very powerful Monitor and Control framework that makes the life of our software engineers much easier. Development of software devices in Python 3 became a simple task. Rapid prototyping with quick cycles of testing, debugging, bug fixing showed to be extremely valuable not only for software-only devices but also for devices that represented parts or all of one of our Hardware Translators. Especially when hardware was involved, our Attribute Wrapper has shown its incredible power due to its built-in separation of Tango Controls on one side and interfacing with hardware on the other side.

Without Tango Controls, our Attribute Wrapper, OPC UA and the SKAO Docker images for Tango Controls, we would not have been able to deliver the wealth of functionality with the little manpower that we had at our disposal.

The entire source code of the LOFAR2.0 Station Control project [17] is made publicly available under the Apache, Version 2.0, Open Source license [18]. The SKAO Docker images that we use are publicly available at the SKAOś artefact repository [19].

REFERENCES

[1] M. van Haarlem, M. P. Wise *et al.*, "LOFAR: The LOw-Frequency ARray", in *Astronomy & Astrophysics*, vol. 556, 2013, p. A2. doi:10.1051/0004-6361/201220873

[2] P. C. Broekema, J. J. D. Mol, *et al.*, "Cobalt: A GPU-based correlator and beamformer for LOFAR", in *Astronomy and Computing*, vol. 23, 2018, pp. 180–192. doi:10.1016/j.ascom.2018.04.006

[3] H. W. Edler, F. de Gasperin, and D. Rafferty, "Investigating ionospheric calibration for LOFAR 2.0 with simulated observations", in *Astronomy & Astrophysics*, vol. 652, 2021, p. A37. doi:10.1051/0004-6361/202140465

[4] G. W. Schoonderbeek, A. Szomoru, *et al.*, "UniBoard^2, A Generic Scalable High-Performance Computing Platform for Radio Astronomy", in *J. Astron. Instrum.*, vol. 08, no. 02, 1950003, 2019. doi:10.1142/S225117171950003X

[5] W. Mahnke and S. H. Leitner, "OPC Unified Architecture - The future standard for communication and information modeling in automation", in *ABB Review*, vol. 3, 2009, pp. 56–61.

[6] J. M. Chaize, A. Götz, W. D. Klotz, *et al.*, "TANGO - An Object Oriented Control System Based on CORBA", in *Proceedings of the 7th ICALEPCS (ICAPELCS'99)*, 1999, paper WA2I01, pp. 475–479.

[7] Object Management Group, "CORBA: Common Object Request Broker Architecture", https://www.omg.org/spec/CORBA/

[8] S. Rubio-Manrique, T. Coutinho, and R. Suñé, "Dynamic Attributes and Other Functional Flexbilities of PyTango", in *Proc. ICALEPCS'09*, 2009, paper THP079, pp. 824–826.

[9] Square Kilometre Array Observatory, https://www.skao.int/

[10] ELK stack, https://elastic.co/what-is/elk-stack

[11] Prometheus, https://prometheus.io

[12] Grafana, https://grafana.com

[13] Jupyter Notebook, https://jupyter.org

[14] M. Di Carlo, P. Harding, *et al.*, "TANGO-grafana: an online diagnostic tool to assist in the analysis of interconnected problems difficult to debug in the context of the Square Kilometre Array (SKA) telescope project", in *Proceedings of the SPIE*, vol. 11452, Software and Cyberinfrastructure for Astronomy VI, 2020. doi:10.1117/12.2576297

[15] Sphinx Python Documentation Generator, https://www.sphinx-doc.org

[16] Read the Docs, https://readthedocs.org

[17] Git repository for ASTRONś LOFAR2.0 Station Control source code,https://git.astron.nl/lofar2.0/tango

[18] Apache Licence, Version 2.0, https://www.apache.org/licenses/LICENSE-2.0.html

[19] Square Kilometre Array Observatory: Docker images for Tango Controls, https://artefact.skao.int/#browse/browse:docker-all:v2

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

THE ELT CONTROL SYSTEM: RECENT DEVELOPMENTS

G. Chiozzi, L. Andolfato, J. Argomedo, N. Benes, C. Diaz Cano, A. Hoffstadt Urrutia,
N. Kornweibel, U. Lampater, F. Pellegrin, M. Schilling,
B. Sedghi, H. Sommer, M. Suarez Valles
European Southern Observatory, Garching bei Muenchen, Germany

Abstract

The Extremely Large Telescope (ELT) is a 39m optical telescope under construction in the Chilean Atacama desert. The design is based on a five-mirror scheme, incorporating Adaptive Optics (AO). The primary mirror consists of 798 segments with 1.4m diameter. The main control challenges can be identified in the number of sensors (~25000) and actuators (~15000) to be coordinated, the computing performance and small latency required for phasing of the primary mirror and the AO. We focus on the design and implementation of the supervisory systems and control strategies. This includes a real time computing (RTC) toolkit to support the implementation of the AO for telescope and instruments. We will also report on the progress done in the implementation of the control software infrastructure necessary for development, testing and integration. We identify a few lessons learned in the past years of development and major challenges for the coming phases of the project.

INTRODUCTION

The ELT is a large segmented telescope, where significant wavefront perturbations are induced by the telescope itself (deformation through gravity, temperature, and wind loads), in addition to perturbations added by the atmosphere. The goal is to control the telescope enabling the delivery of a diffraction limitable beam at each of the ELT Nasmyth foci, i.e. where the light beam is transferred to the instruments. This means the "spectrum of wavefront aberrations induced by the observatory is below that of the free atmosphere." [1].

The ELT Control System implements the overall control of the telescope (and dome), including the computers, communication and software infrastructure. It defines standards for control and electronics hardware and software and data communication. It includes the high-level coordination software, wave front control computer and engineering data archive.

In a system the size of the ELT Control System, decisions on algorithms and computational performance are not the only major design problems. The organization of the overall system, its behavior and interactions represent a significant organizational complexity which must be addressed.

An important factor influencing the architecture of the control software is the procurement strategy, that foresees the outsourcing of all components and services which can be efficiently delivered by industrial partners, while maintaining in-house those tasks for which ESO has a particular domain expertise. Based on this principle, also the overall control system shall be composed of components designed, built and delivered by many industrial partners or in-house. Distributed development and integration of the subsystems demand clear interfaces which should match not only a functional breakdown of the control system, but reflect the organizational boundaries of the many development locations.

CONTROL STRATEGY

The main challenge of ELT is to provide a wavefront with an error in the range of 10th of nm in the presence of perturbations that can be in the range of mm (in the case of gravity deformation when changing the telescope pointing from zenith to horizon). The most important role in the associated control strategy is played by the deformable quaternary mirror (M4). It is controlled in an on-sky loop using stellar light with a large temporal and spatial bandwidth. This requires a deformable mirror of unprecedented size. M4 has 5352 degrees of freedom, with the on-sky loop being closed at rates up to 1 kHz [2]. The limited stroke (100um) of the M4 actuators and the limited capture range of the wavefront sensors exclude that the wavefront can be controlled solely by M4, but require it to be supported by several additional control systems:

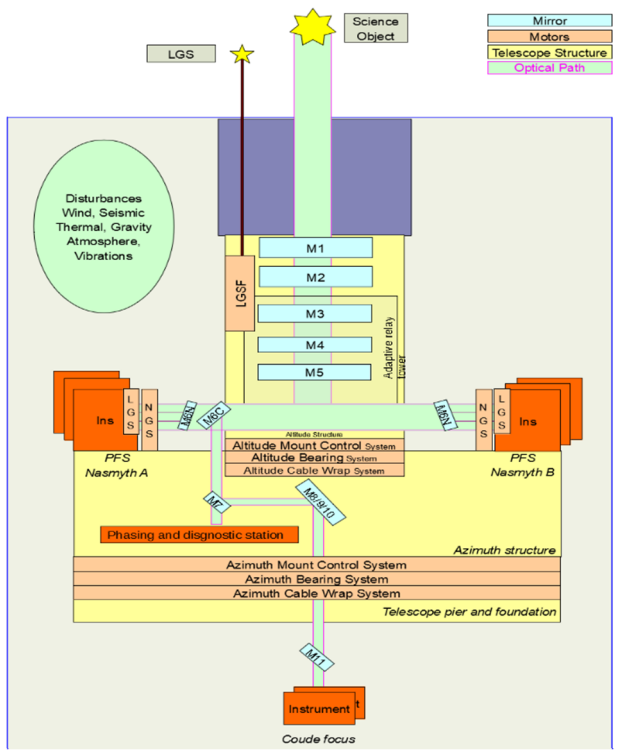


Figure 1: Telescope subsystems following the light path.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Feed-forward control is used mainly during the "blind phases" of the Telescope when on-sky loops are open. E.g. a pointing system is used to preset the telescope to a new target, with the feed forward model taking into account astrometry and telescope deformations due to gravity and temperature to bring the telescope within the acquisition range of on-sky sensors [3].
- Feedback loops based on telescope internal metrology (as opposed to on-sky loops) are used to control the state of the telescope. E.g. the large segmented primary mirror consisting of 798 hexagonal segments, and the M1 control system moves each segment in piston, tip, and tilt (2394 degrees of freedom) to control the shape of M1 based on measurements of relative displacements between segments using 4524 Edge Sensors. The M1 Figure Loop can reduce relative edge displacements to several nm and can keep low order deformations (in the mm range due to gravity) at levels that are within the capture range of M4 [4].
- Stroke Management is a background task to redistribute the slowly building non-zero-mean components in low order modes of M4 to other degrees of freedom:
  - tip and tilt modes are controlled in a collaborative control scheme together with M5 and Telescope Main Axes, which desaturates M4 eventually through a Main Axes guide correction. This process is referred to as Field Stabilization [5,6].
  - focus and coma are transferred to M2 occasionally, at most every 5 minutes [7].
  - higher orders are continuously offloaded to M1, essentially by commanding the low pass filtered modal amplitudes accumulated on M4 to the M1 figure loop control system.

Moving optical surfaces changes the wavefront error measured by the Adaptive Optics system and causes the M4 to desaturate, i.e. saturation management requires a closed on-sky loop. Stroke management redistributes stroke between the optics degrees of freedom but does not aim on maintaining the telescope at its prescription [7].

While the control strategies outlined above are simulated [8] and tested [9,10,11] in detail, it is important to note that the unprecedented size of ELT is expected to lead to surprises during commissioning. Changes in the control strategy are hence expected, and therefore the control system is built such that well understood subsystem control is decoupled from less defined high-level (on-sky) control, and the latter one is developed in a way that allows for the flexible adjustment of algorithms during commissioning.

CONTROL SYSTEM ARCHITECTURE

The ELT Control System architecture has been described in [12] and since then we have consolidated and further developed it in the details, using the feedback from prototypes and the first applications. Here and in the next sections we will describe just some key aspects to have a context and we will describe the most important developments since the publication of [12].

The architecture enforces a distinction between a telescope device (*the System Under Control - SUC*, e.g. the M2 unit) and the component controlling that device (*the Control System - CS*). This terminology is adopted from the State Analysis (SA) methodology developed at JPL [13].

As a system of systems, the ELT contains layers of controllers. A lower level component comprising a CS and SUC appears as a SUC to a higher-level CS. For example, the primary mirror segment position actuators (PACT) with embedded position controller and course and fine stage actuators appear as components of the SUC to the M1 CS responsible for figure control.

Figure 2 shows an overview of the ELT CS, with some simplifications and omissions for readability (numbers circled in orange are used to identify items referenced with Fig. 2-# in the text below).

The first breakdown of the ELT CS is into the many individual control systems associated with Telescope subsystems (called the Local Control Systems (LCS)) (Fig. 2-1), and the single system that integrates these, termed the Central Control System (CCS) (Fig. 2-2), whose internal structure is described in the corresponding section below.

The LCS-CCS discrimination not only separates unit-level from telescope-level safety and control, it also matches organizational boundaries in-line with the ELT procurement strategy: individual subsystems (mirror units, main structure, …) are designed, built and delivered by industrial partners; their integration is ESO responsibility.

On the other side of CCS are instruments (Fig. 2-3), developed by Consortia of ESO partner institutes. Each instrument includes an independent Instrument Control System (ICS) developed following the ELT standards, and interfacing with the telescope through the CCS interface, split over a control and a deterministic network (Fig. 2-4).

The interfaces between LCSs, CCS, and instruments are defined in a series of ICDs specifying the logical addresses, data types, formats, rates and characteristics of the data communication. The same applies to the interfaces between the internal components of CCS, even if that is fully developed inside ESO. This strict interface management is key, given the distributed nature of the ELT CS and the range of developers and suppliers.

From State Analysis we have also adopted the *State Variable* and *Estimator-Controller-Adapter* patterns[13].

The term *State Variable (SV)* refers to an element of the CS that represents a physical state of the SUC. For example, a limit switch in the SUC will physically be in an opened/closed state and the CS will use evidence such as sensor measurements to *estimate* the state of the switch as opened/closed. SVs are observable by clients.

The *Estimator-Controller-Adapter* pattern is based on:

- The Adapter that allows to communicate with the SUC to command and measure.
- The Estimator that receives measurements from the SUC via the Adapter and computes State Variables. Estimators do not send commands to the SUC.
- The Controller that is responsible to control the SUC via the Adapter. It can access State Variables if needed.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

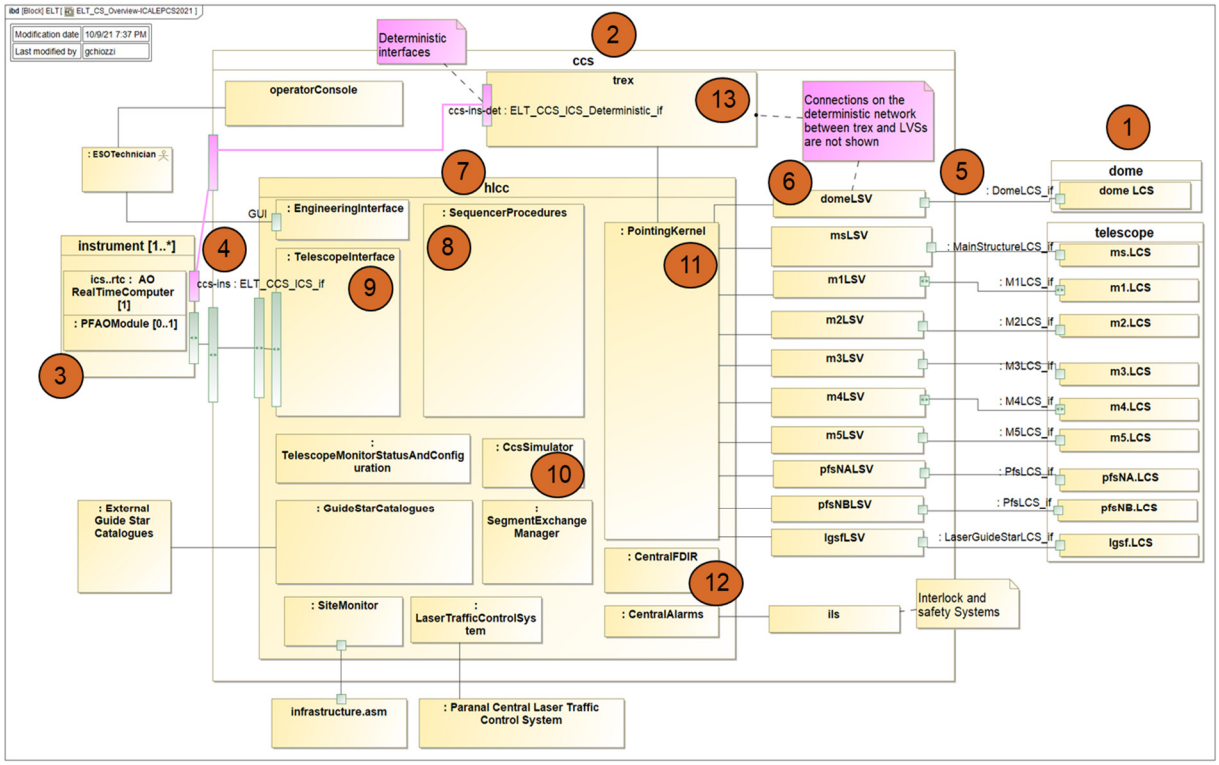


Figure 2: ELT Control System overview.

Commands sent by a Controller affect the CS and therefore the value of State Variables.

We have decided (after analysis and prototyping) not to adopt *SA goal-based operation*. This would have changed radically our way of viewing the system with respect to previous projects and the experience of the astronomers developing observations. Since we do not have the requirement to develop an autonomous system (where goals-based operation is a major advantage), we believe that the cost of a paradigm shift would have not been compensated by the benefits.

LOCAL CONTROL SYSTEMS

A Local Control System (LCS) contains two key functional groups: control and safety. Control functions enable standard operations, while safety functions preserve integrity and guarantee safety of personnel and equipment.

The control functions include Control Software, Local Control Unit(s), remote IO and a local communication.

The safety functions include Safety Logic, Local Safety Unit(s), Safety IO devices and a fail-safe communication.

The LCS enables safe control of the functions of the associated subsystem (e.g. M2 mirror cell). The functions provided by the LCS can make no assumptions as to the nature of the subsystem use in the context of the telescope control system (the operation and wave front control strategies being implemented by the CCS). For example, the M4 adaptive mirror has to be available and operable to full performance irrespective of whether or not the telescope is observing or calibrating or parked.

The LCS provides interfaces to CCS that enable individual and independent control of subsystem devices and functions (Fig. 2-5). For example, it must be possible to move the warping harness of an M1 segment irrespective of the status of edge sensors and position actuators.

The common characteristics of LCSs and the specific example of the M1 LCS, that is developed directly at ESO as well as the Laser Guide Star (LGS) LCS, have been described with more details in [14]. All other LCSs are being implemented by the contractor responsible for each individual subsystem; at present time, most telescope subsystems have passed final design review and are in different stages of implementation. Each contractor is followed-up by a member of the ESO CCS development team. Particularly important is the definition and consolidation of the ICDs between each LCS and CCS as the detailed design and implementation progress.

CENTRAL CONTROL SYSTEM

The Central Control System (CCS) integrates the many Local Control Systems into a single system implementing the coordinated control, system level safety, monitoring and user interfaces required to operate the telescope.

It provides monitoring, logging and archiving for long term trending and configuration control. Control Room terminals, GUIs and tools belong to CCS. CCS is therefore the primary user interface to operate the Telescope.

CCS (developed in-house, to leverage our specific astronomical expertise) is responsible for coordination and for all what concerns the astronomical domain.

CCS applications are organized in a shallow hierarchy of loosely coupled cooperating components as can be seen in (Fig. 2-2). All applications are based on a common software infrastructure (see [12] and the section below) and

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

in particular on the Rapid Application Development (RAD) framework [15]. RAD helps in the development of event driven applications by imposing a common design and providing tools to quickly produce application skeletons ready to use. RAD applications are built around a BOOST ASIO event loop [16] integrated with a SCXML state machine interpreter [17]. Anonymous publisher-subscriber communication and shared access to an Online Database are used to keep the coupling as loose as possible.

LSVs

Within CCS, a Local Supervisor (LSV) (Fig. 2-6) is a software component that provides access to specific Local Control System (LCS) functionalities. The LSV is responsible for implementing the telescope domain logic and translating the telescope concepts into the device domain handled by the corresponding LCS. For example, the M1 LSV is responsible for controlling and maintaining a certain optical quality of the whole telescope primary mirror surface while the M1 LCS is managing the actuators and sensors installed on each segment of the mirror.

There is one LSV for each of the following telescope LCSs: M1, M2, M3, M4, M5, Pre-Focal Stations, Main Structure, Dome, and Laser Guide Star. Each is independent from the other LSVs and can use only the services of the corresponding LCS. For instance, M1 LSV is not allowed to communicate directly to M4 LSV and can send commands only to M1 LCS.

The services provided by an LSV are grouped in "subsystem functions". Each subsystem function corresponds to a set of functionalities implemented by the LCS. For example, since the Main Structure LCS allows to position the altitude and azimuth axes independently, the Main Structure LSV provides two subsystem functions, one to deal with altitude and one for azimuth. For each subsystem function an operational state is estimated by the LSV using the published LCS measurements. The subsystem functions are supposed to be, as far as possible, independent (e.g. it should be possible to use altitude axis even if azimuth is not available and vice-versa).

Each subsystem function is made of an adapter library, one or more estimator applications, and zero or more controller applications following the Estimator-Controller-Adapter design pattern. Each estimator and each controller are implemented by a dedicated application. LSV estimator and controller applications share the same architecture to allow faster development and easier maintenance.

Common requirements across all LSVs and specific requirements have been identified, a set of common design patterns has been drawn and implemented in pathfinder LSV applications. Detailed design for the individual LSVs is being analyzed and the development has started and will be the main activity for the coming years until deployment in the ELT Control Model[12] and at the telescope.

HLCC

The High-Level Coordination and Control (HLCC) software layer lies above the LSVs (Fig. 2-7). It offers a single interface of the whole telescope toward operators and the instrument control software. Its main task is for supervisory applications to coordinate the various telescope subsystems.

The main challenge for HLCC is to implement a well-structured system that at the same time can be modified to a large extent during telescope commissioning.

To reach this objective we have identified the building blocks that can be seen in Fig. 2. An important role will be played by the SequencerProcedures (Fig. 2-8). These will be developed around *features*[12], independent supervisory applications designed to perform a complete operational function/use case of value to the users of the system.

We have planned for a long period of integration and commissioning[18], during which we will discover how to operate our machine and how the elementary functions provided by the LSVs will have to be composed together. Implementing *features* as independent components, using an interpreted language (Python) accessible to the commissioning team (not necessarily SW developers) allows us to evolve them in an easy way, with minimal impact on other *features*.

In the last two years we have developed a prototype to validate our architecture and design with respect to the software infrastructure and the application framework. We chose a vertical slice from a dashboard GUI down to several services representing a telescope tracking the sky. Here we give a few examples of design choices that differ from existing software at other ESO telescopes.

Measurement data was treated using estimators subscribing to the publishers that deliver input data to them, processing this data in Java code or in external Python scripts (Jep framework[19]), and pushing results out through StateVariables. Estimators form a hierarchy, e.g. with 2 estimators for incoming Alt and Az positions of the telescope, and downstream estimators that combine or convert this data. The use of pub-sub communication makes deployment of estimators in one or many processes a flexible choice. We got smooth data flow through these estimators when feeding them with simulated data at 20 Hz. One of the estimators for derived data produced the telescope's actual state, e.g. as "tracking" or "moving".

The first HLCC prototype of the RAD applications was implemented in Java, that we considered better suited than C++ for high-level coordination, without demanding performance requirements.

For the current implementation of the HLCC applications, we decided to change from using Java and Python to using C++ and Python. Java worked very well for the prototype, and likely would have worked well also for the final applications. But LSV and instrumentation applications are being developed in C++ to leverage the huge experience in the ELT software development team and to avoid potential performance problems coming from JVM garbage collection, which could introduce unwanted jitter to applications. We hope that the lower efficiency we experience working with C++ will be compensated by synergies and code reuse across subsystems as well as less maintenance in the lower level infrastructure software through decreased support for the Java language.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

We identified the HLCC interfaces and applications in a first design iteration and we have started now with the implementation. The Telescope Interface implements the ICD between all of CCS and the instruments. All requests from Telescope Interface will be served by delegating to other HLCC or LSV applications or to the Telescope Real-time Executor (TREx), described below.

As an alternative, especially during early development when these other applications do not yet exist or do not yet provide simulation capabilities of their own, the TelescopeInterface (Fig. 2-9) will delegate to the CcsSimulator (Fig. 2-10). The CcsSimulator will be also delivered to the consortia developing instruments, to have a frontend for testing their interactions with CCS.

In addition to the request-reply and publish-subscribe communication described in the ICD, instruments, as well as operators, will be able to access dedicated telescope data published in the Online Database.

The PointingKernel (Fig. 2-11) application controls all telescope subsystems. It interacts with TREx for most of the pointing logic, but also commands the LSVs directly.

The CentralFDIR (Fig. 2-12) application monitors those aspects of quality and failures that involve more than a single subsystem.

Other dedicated applications exist for star catalogues, monitoring and configuration, alarms, or specific tasks such as segment exchanges.

TREx

Control loops with demanding real-time requirements described in the Control Strategy section are not part of HLCC; instead, they are allocated to the Telescope Real-time Executor (TREx) (Fig. 2-13), which communicates directly with LSVs and LCSs using, when necessary, the deterministic network (not shown in the figure). HLCC commands and monitors TREx.

This component of the system is now in the requirements' collection phase. What is clear is that the WFC and stroke management strategy and algorithms will, for a big part, be developed as part of commissioning, while gaining experience on the as-built telescope.

For this purpose, a flexible software framework for real-time control applications is required. The framework must be suitable for use by control engineers and a visual programming environment, comprising a palette of data analysis and algorithmic building blocks, is foreseen. The framework should not require detailed knowledge of real-time systems when mapping the application to the underlying hardware resources.

A demonstration prototype was developed in 2019 based on GNU Radio[20], a free open-source development toolkit to implement signal processing tasks. Although GNU Radio is not built with low-latency and control systems in mind, it seems possible to overcome the limitations with minor adaptions and by tailoring our use to the pure algorithmic domain. CS specific functions (e.g. I/O, monitoring, error handling and recovery) should be handled in a separate entity that interfaces the real world with GNU Radio. GNU Radio comes with a GUI usable by non-software engineers and integrates well with C++ and Python.

Results with a computational load comparable to TREx show that 1kHz loop rates are reliably achievable. Measurements indicate that 2kHz loop rates may be possible.

RTC AND ADAPTIVE OPTICS

The core component of any AO system is the Real Time Computer (RTC) which measures the incoming wavefront aberrations by means of sensors and corrects for them by means of a deformable mirror.

During scientific observation, the ELT telescope performs only guide probe AO, i.e. measurements done with a wavefront sensor (WFS) installed on the pre-focal station (PFS) and limited to the first few modes are used to reject low and mid spatial/temporal frequency wavefront. This is required to provide an image quality sufficient for handover to an instrument. Guide probe AO is performed by TREx, which stops performing it after INS handover; at this point the instrument drives the M4 to achieve the desired image quality [7].

There will be therefore one RTC per each instrument to implement high order AO modes and one RTC used on the telescope for commissioning and diagnostic (the Phasing and Diagnostic Station).

The ELT RTC architecture [21] is defined by ESO not only for the telescope, but also for the instruments, with the aim of streamlining development and leveraging re-usability. This architecture identifies two distinct components (Fig. 3).

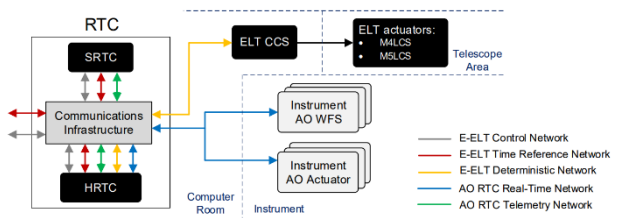


Figure 3: RTC architecture.

Each of them targets functions in a specific domain and timescale and follows its own technology roadmap.

The Hard Real-Time Core (HRTC) implements the main AO control loops, which perform demanding computations on incoming WFS measurements and command actuators within tight timing constraints. The HRTC is interfaced to the AO WFSs and actuators on the Instrument via a dedicated RTC Real-Time Network. The HRTC commands both the M4 and the M5 via the Deterministic Network.

The Soft Real-Time Cluster (SRTC) is a set of computing nodes in charge of the high-level supervision and optimization of the HRTC. It is driven by requests from Instrument Control Software (ICS), as well as by the reception and automatic processing of telemetry data (e.g. measurements, commands). Computations elapse from seconds to minutes and involve algorithms operating on large data sets. A dedicated AO RTC Telemetry Network

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

interconnects the HRTC and SRTC. ESO has defined standards for the SRTC technology and delivers the AO RTC Toolkit: common RTC functions are addressed by a suite of software tools, libraries and reference implementations.

The HRTC is on the cutting edge of technology, and therefore rather than standardizing technology, we have decided to specify only interfaces, to achieve full replaceability as mitigation of obsolescence.

Nonetheless, the HRTC prototype (HRTCp) has been developed to explore the feasibility of building an ELT-size, AO RTC based on mainstream CPU and Ethernet technology - i.e. without accelerators or specific-purpose hardware. Key aspects to be assessed are modularity, scalability, long-term maintainability and upgradeability.

The HRTCp addresses a Multi-Conjugate AO (MCAO) configuration with six LGS WFS and adopts a functional breakdown that enables scaling it down to Single Conjugate AO (SCAO), such as the ELT Wavefront RTC (WFRTC), i.e. the RTC to be used in combination with the PFS and during commissioning for verification of telescope SCAO capabilities. The HRTCp operates at 500 Hz and implements a pseudo open-loop control (POLC) algorithm dominated by two memory-bound, Matrix-Vector-Multiplication (MVM) operations of size 6,316x55,392 per loop cycle. All the external interfaces use 10 GbE signaling to ingest an aggregate 44.4 Gbps incoming pixel traffic and produce 87 Mbps actuator commands and 2.7 Gbps telemetry data towards the SRTC. Simultaneously, the system continuously receives disturbance data at 1 Gbps.

The *as built* HRTCp is comprised of 14 mainstream servers: two front-end nodes (namely A and B) per LGS WFS, one shared back-end node and one low-level supervisory node. Front-end node A implements the MVM in the direct control path using 48 CPU cores on a dual-socket AMD EPYC 7742 server. Front-end node B is responsible for the MVM in the offline control path (burst computation) and employs 56 cores on a dual-socket AMD EPYC 7501 server. Aggregate, each pair of front-end nodes delivers in excess of 600 GB/s memory bandwidth and 300 GFLOP/s only for these operations.

The internal communications employ 10 GbE, except for the collection of results from the front-end A nodes by the back-end node. Early tests showed that this is a major serialization point (contributing a 100 us delay over 10 GbE), thus dictating the use of single 100 GbE link. With this, the current HRTCp end-to-end latency (measured from last pixel image packet received to last command packet sent) is 244 us. The associated deviation is below 4 us over 2-minute intervals, with maximum excursions of 25 us.

The key challenge has been to achieve the required performance using only hardware platforms and software techniques with long-term maintainability and upgrade paths. A key mandate was to widen the profiles that could contribute to RTC development by targeting knowledge domains (i.e. CPU-based computation and Ethernet networking) available at ESO.

The development was started in early 2018 and the system is currently in operation at ESO premises. Important picks from this prototype development are:

- Recent, mainstream CPU architectures targeting High Performance Computing and providing fine NUMA granularity can be leveraged to fit our problem.
- Deterministic networking can be achieved with the native Linux network stack, using interrupt routing and controlling packet coalescence. Deterministic, overall latency performance is possible exploiting real-time techniques such as NUMA affinity, core isolation, thread pinning and inter-thread polling.
- It is possible to write maintainable C++ software that implements the above techniques, with little or no explicit vectorization (i.e. offloading this to the compiler). The current trend for increased CPU core count in CPU families helps mitigate the architectural constrains derived from core isolation and thread pinning.

The HRTCp will be used as a flexible platform where upcoming CPU families and networking devices can be benchmarked. In addition, several of the HRTCp design choices will be at the core of the ELT WFRTC.

A subset of the techniques developed within the HRTCp scope have been ported to the VLT domain. An upgrade of the VLT SPARTA systems will replace the obsolete, FPGA-based real-time core with a single server, while respecting the legacy (non-Ethernet) I/O interfaces.

THE MINUSCULE ELT (MELT)

The Miniscule ELT (MELT) [9] is an optomechanical test bench comprised of key components such as a segmented primary mirror, a secondary mirror on a hexapod, an adaptive fourth mirror, and a fast tip/tilt mirror together that mimic certain functionalities of the ELT.

It is meant for testing and validating key functionalities to be used on the ELT during system verification, wavefront control commissioning, through the handover to science, up to regular diagnostic, monitoring, or validation during operations.

The main objectives of MELT are to deploy and validate the telescope control system as well as wavefront control algorithms for commissioning and operations.

The optomechanical setup uses the Active Segmented Mirror (ASM) with 61 piezo-driven segments and a diameter of 15 cm. It was used on sky on a VLT telescope during the Active Phasing Experiment (APE)[22].

Several beam paths after the optical train on MELT are conditioned and guided to wavefront sensors and cameras, sensitive to wavelength bands in the visible and infrared to emulate wavefront commissioning and phasing tasks.

In MELT, the ELT main axis control is emulated with a moveable diffraction-limited source that emits white light from the visible up to the K band through a turbulence generator. A single conjugate adaptive optics Shack Hartmann (SH) WFS is used in closed loop with an ELT RTC and M4 to test and validate offloading scenarios to M5 and the main axis. In addition, it is used to deploy and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

validate wavefront control algorithms and the influence of AO on M1 phasing using the baseline SH high order WFS, but also M4 phasing issues with its petals, and scalloping. The bench also allows to test different phasing concepts.

The MELT Control System applies the same architectural concepts as the ELT. Furthermore, the central services developed for the ELT, are deployed in MELT as soon as they are released, such that their usage provides early feedback. MELT is therefore an excellent testbed also for the whole control SW architecture and tools.

The bench will help us to be as much as possible prepared when the telescope will send the star light through the optical train to be able to tackle the unforeseeable problems and not be caught up with the foreseeable ones.

COMMON SW INFRASTRUCTURE

The control software of the ELT is built on top of a common software infrastructure, comprising libraries, frameworks and development tools for building, deploying, documenting and testing the applications.

Among these, particularly important is the Core Integration Infrastructure (CII). CII comprises communication libraries, configuration infrastructure, network value cache, and logging infrastructure for control applications. The communication part includes an interface definition language for defining the request/reply interfaces as well as the publish/subscribe contracts, independent from the actual protocol (ZMQ, DDS, OPC/UA, and others) used. CII provides APIs in three languages (C++, Python, and Java).

CII relies on a number of third-party products. For example, the Online Database (a central value cache on the control network) keeps its data split over three third-party databases: elasticsearch, redis, minIO. While we benefit from the stability and feature richness of these products, mastering them is a challenge. We also often find the need of creating additional tooling, in particular when we aim to allow our users to handle them on their development set-ups without direct support from the CII team.

Navigating in the problem space of performance versus usability versus maintainability, we are frequently discussing whether a given usage pattern should be supported in the CII infrastructure layer or not. A clear cut would be desirable between functionality implemented in ground-layer infrastructure, versus middle-layer frameworks, versus higher-layer subsystems. Defining this cut, or clear criteria for it, is an on-going challenge.

CII has been developed by Cosylab for ESO, from a specification of about 800 requirements in three years; the first version is being adopted by the control subsystems.

Different subsystems have different usage patterns for using the CII software, and currently the main activity consists in adjusting functionality according to user feedback and moving features between software layers.

We have recently transferred the user documentation from documents (word / pdf) to a collaborative platform (GitLab / reST / Sphinx / Jenkins). This enables users to prepare and propose improvements, while still being governed by a well-defined process. This has well decreased the turn-around time for documentation updates and makes our documentation more helpful.

While CII gets more and more used in the control subsystems, we are getting ready to deal with the resulting higher amount of user feedback and support requests.

The development of GUIs is addressed instead by the Control UI Toolkit (CUT). CUT is a set of libraries, widgets and graphical design patterns tailored to the ELT Control System requirements. Qt is used for graphical rendering; Taurus [23,24] as an abstraction layer that provides a powerful MVC pattern specifically designed with control systems in mind. Additional custom widgets, utilities, color schemes, and documentation based in Taurus and Qt complete the GUI development environment.

Taurus was selected due to similarities in requirements to our own specification, offering:

- Extension capabilities: plugins for Taurus can be developed to enhance its communication capabilities, model access, plotting and image rendering.
- MVC design: allows decoupling widget and views development from models. While we develop models to access CII services, we can still continue development of application views using the "eval" model plugin, or any other plugin.
- Multiple expertise levels: developers in ELT varies in GUI software development experience. Taurus offers three ways to develop GUIs, from a very simple no-code approach, to complete freedom [24].
- Declarative binding: supports declaration in UI specification of binding between datapoints and widgets.
- Subscription, polling, filtering, customization and declarative configuration of UI.

Taurus is extensible by design, a fact that is appreciated and allows us to provide support for CII communication libraries and infrastructure as they are integrated in the development environment. At this moment, ELT includes Taurus in its Software Development environment, and we contribute our bug fixes and development directly to the upstream project. It also includes the Taurus CII Online Database Plugin. The Control UI Toolkit has been used until now to implement prototype and engineering GUIs.

CONCLUSION

The ELT Control System faces major challenges that are expected to be overcome by the development of a System of Systems flexible up to the commissioning phase. After the requirements and design phase and the development of the technical infrastructure and of (large scale) prototypes, we are now moving to the serial development of the actual system components. The LCSs, primarily developed externally by outsourced contractors, LSVs and HLCC, shall be integrated into one single System user interface. Specific validation test benches (MELT) are operational and following the evolution of the project.

The Scientific First Light for the ELT is foreseen for the end of 2027 and our schedule is in line with this target.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] H.Bonnet *et al*., "Adaptive optics at the ESO ELT," Proc. SPIE 10703, Paper 10703-10, 2018.

[2] R.Biasi *et al*., "E-ELT M4 adaptive unit final design and construction: a progress report", Proc. SPIE 9909, Paper 9909-7Y, 2016.

[3] P.T.Wallace, "A rigorous algorithm for telescope pointing," Proc. SPIE 4848, 2002.

[4] B.Sedghi, M.Müller, "Dynamical aspects in control of E-ELT segmented primary mirror (M1)," Proc. SPIE 7733, Paper 7733-2E, 2010.

[5] B.Sedghi *et al*., "Field stabilization (tip/tilt control) of E-ELT", Proc. SPIE 7733, Paper 7733-40, 2010.

[6] B.Sedhi, "Tip/tilt control strategies at ELT" Wavefront sensing and control in the VLT/ELT era, 3rd edition, invited talk, Paris, 2018 https://indico.obspm.fr/event/56/contributions/145/

[7] H.Bonnet *et al*., "Adaptive optics at the ESO ELT", Proc. SPIE 10703, Paper 10703-10, 2018.

[8] B.Sedghi *et al*., "E-ELT modeling and simulation toolkits: philosophy and progress status", Proc. SPIE 8336, Paper 8336-06, 2011.

[9] T.Pfrommer *et al*., "MELT: an optomechanical emulation testbench for ELT wavefront control and phasing strategy", Proc. SPIE 10700, Paper 10700-3F, 2018.

[10] M. Dimmler *et al*., "E-ELT M1 test facility", Proc. SPIE 8444, Paper 8444-1Y, 2012.

[11] H.Bonnet *et al*., "Fast optical re-phasing of segmented primary mirrors", Proc. SPIE 9145, Paper 9145-1U 2014.

[12] G.Chiozzi *et al*., "The ELT Control System", Proc. SPIE 10707, Paper 10707-31, 2018.

[13] M.Ingham *et al*., "Engineering Complex Embedded Systems with State Analysis and the Mission Data System," Proceedings of First AIAA Intelligent Systems Technical Conference, 2004, https://trs.jpl.nasa.gov/handle/2014/38225

[14] L. Andolfato *et al*., "The ELT M1 local control software: from requirements to implementation, Proc. ICALEPCS2019, New York, USA, 2019.

[15] ELT ICS Rapid Application Development (RAD), http://www.eso.org/~eltmgr/ICS/documents-latest/RAD/sphinx_doc/html/

[16] Boost asio, https://www.boost.org/doc/libs/1_77_0/doc/html/boost_asio.html

[17] State Chart XML, https://www.w3.org/TR/scxml/

[18] R.Tamai *et al*., "The ESO's ELT construction progress", Proc. SPIE 11445, Paper 114451E-16, 2020.

[19] JEP – Java Embedded Python, https://github.com/ninia/jep

[20] GNU Radio, https://gnuradio.org

[21] M. Suárez Valles *et al*., "Adaptive Optics Hard and Soft Real-Time Computing Developments at ESO," AO4ELT6 Adaptive Optics for Extremely Large Telescopes, Quebec, June 9-14, 2019.

[22] F.Gonte *et al*., "APE: the Active Phasing Experiment to test new control system and phasing technology for a European Extremely Large Optical Telescope," Proc. SPIE 5894, Paper 5894-0z, 2005.

[23] C.Pascual-Izarra *et al*. "Taurus big & small: from particle accelerators to desktop labs," in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 166-169. doi:10.18429/JACoW-ICALEPCS2017-TUBPL02

[24] C.Pascual-Izarra *et al*., "Effortless creation of control & data acquisition graphical user interfaces with taurus," in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, pp. 1138-1142. doi:10.18429/JACoW-ICALEPCS2015-THHC3003

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL02

REAL-TIME FRAMEWORK FOR ITER CONTROL SYSTEMS

W. Lee†, A. Zagar, B. Bauvir, T. Tak, ITER Organization, St. Paul Lez Durance Codex, France
A. Winter, Max Planck Institut für Plasmaphysik, Greifswald, Germany
M. Knap, P. Karlovsek, Cosylab d.d., Ljubljana, Slovenia
S. Lee, Korea Institute of Fusion Energy, Daejeon, Republic of Korea
P. Perek, D. Makowski, Lodz University of Technology, Lodz, Poland

Abstract

The ITER Real-Time Framework (RTF) is a middleware providing common services and capabilities to build real-time control applications in ITER such as the Plasma Control System (PCS) and plasma diagnostics.

The RTF dynamically constructs applications at runtime from the configuration. The principal building blocks that compose an application process are called Function Blocks (FB), which follow a modular structure pattern. The application configuration defines the information that can influence control behaviour, such as the connections among FBs, their corresponding parameters, and event handlers. The consecutive pipeline process in a busy-waiting mode and a data-driven pattern minimizes jitter and hardens the deterministic system behaviour. In contrast, infrastructural capabilities are managed differently in the service layer using non-real-time threads. The deployment configuration covers the final placement of a program instance and thread allocation to the appropriate computing infrastructure.

In this paper, we will introduce the architecture and design patterns of the framework as well as the real-life examples used to benchmark the RTF.

INTRODUCTION

The Plasma Control System is a dominant factor for the ITER pulsed operation, it controls all aspects of the plasma discharge from powering the superconducting magnets up to plasma termination [1]. PCS takes data from sensors and applies sophisticated algorithms to generate commands that are sent to actuators to control plasma parameters, such as position, shape or stability in a real-time context. Design, development and verification of real-time software in general is a complex and often lengthy process requiring multiple iterations until all timing relationships are satisfied and the application is stable and predictable.

The RTF is a flexible high-performance software base that facilitates the development and deployment of complex real-time applications [2]. Originally developed with the aim of control algorithms, the RTF can also be the basis for real-time data processing applications in ITER diagnostic systems.

The architecture design fully considered the modularity and portability of the software, and is applicable and extendable even in none-ITER environments. It hides many details specific to real-time systems (e.g., thread management, inter-thread data transfers, etc.), making the design and development of real-time software much easier and faster.

† email address: woongryol.lee@iter.org

Strict Quality Assurance (QA) process and code audits enforced software integrity to bring reliable system operation. Along with the EPICS pvAccess interface that enriches functionality for operation, the Simulink wrapper block allows control model transition from the design to the application in an agnostic way.

ARCHITECTURE

Overview

The RTF infrastructure provides a modular, fully abstracted environment with the following key features [3]:

- FBs are self-contained and do not have any dependency on hardware, inputs, and outputs or operating system within the code. All relevant information for the modules is delivered via configuration, fully reusable in any context.
- Full configurability of FBs, which can be chained together at the developer's discretion by configuration.
- Fully data-driven workflow. The FBs can be scheduled automatically based on the availability of input data.
- Configuration-based distribution of processing logic over different threads, processes and computer nodes (hosts).
- Support integrated operability using generated code from graphical system modelling tools (e.g. Simulink [4]).
- Full integration with ITER Control Data Access and Communication (CODAC). Out of the box support for multiple interfaces to other CODAC components (e.g. networks, archive, supervision, etc.).

Figure 1 shows the architecture of the RTF and a Real Time (RT) application including their main elements and how they interact. The main elements are:

- The **RT application** contains the processing logic that runs on different threads, processes or computer nodes (hosts) and contains:
  - The scheduler handling the execution of processing of FBs.
  - The FBs representing an operation with inputs and outputs.
  - The gateways responsible for ensuring that the data is transported between the FBs running in different threads, processes or nodes.
  - The RT applications running within multiple instances of the real-time process.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

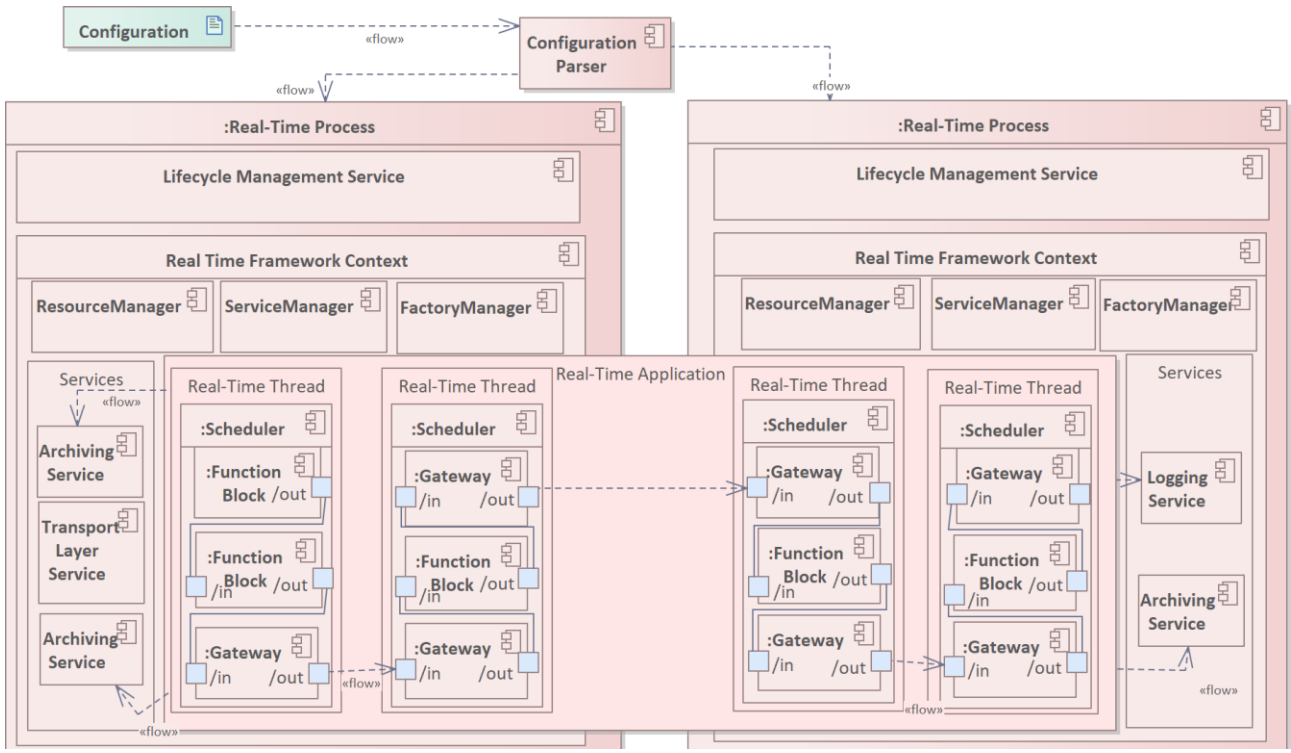


Figure 1: Context diagram of the Real-time Framework.

- The **real-time process** is the main process that runs on a node. This process has been specifically configured for real-time. It contains the following:
  - A life cycle management service managing the life cycle of the framework.
  - The real-time framework context - the key element of the real-time process consisting of Buffer, Queue and Thread Managers that help with tasks like allocating memory, transporting data and executing FBs. RTF context also includes Threads that execute the RT application and Services that provide high-level functionality for the RT application, e.g., logging and archiving.

RT Application

Figure 2 shows a simple RT application that acquires data from two inputs, adds them together and applies them to an output. Each component in Fig. 2 represents a FB, and the links show the connection between input ports (on the left of the component) to output ports (on the right of the component). The framework handles the dependency-based execution of FBs in either single-threaded or multi-threaded environments as specified in the deployment configuration.

Function Block

The Function Block is a representative atomic component to build application. The FB is similar terminology to the Blocks in the Simulink or the Functions in the LabView [5].

The FB is a function responsible for the desired operation, from primitive to complex process algorithms,

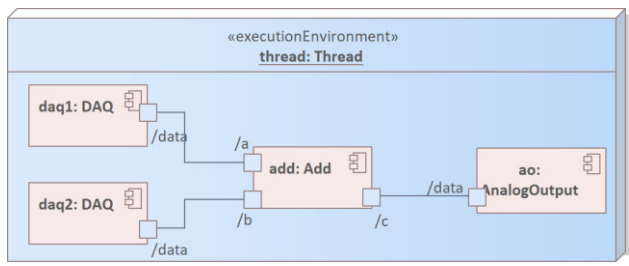


Figure 2: Example of an RT application executing in a single thread.

depending on the design intent. Each FB contains essential interface points: parameters, input signals and output signals that are constructed using a factory design pattern. Additonally it supports event trigger/handler along with prarmeter writers for asynchronous operation shown in Fig. 3.

FBs can be delivered as shared libraries and loaded to RTF-based applications in run-time as dynamic plugins. It allows decoupling user applications from core of the RTF. Designers can develop, maintain and share their custom libraries under different lifecycles and purposes. Only few of common FBs such as reading from and wrting to console or file, and basic mathematical operations are provided with the framework.

FBs can encapsulate other FBs giving the RT application an apparent hierarchical structure. In the case of composite FBs, such encapsulation can either have functional implications on e.g., scheduling or conditional execution of contained FBs, or is used to simply group FBs for entirely conceptual or convenience reasons.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL02

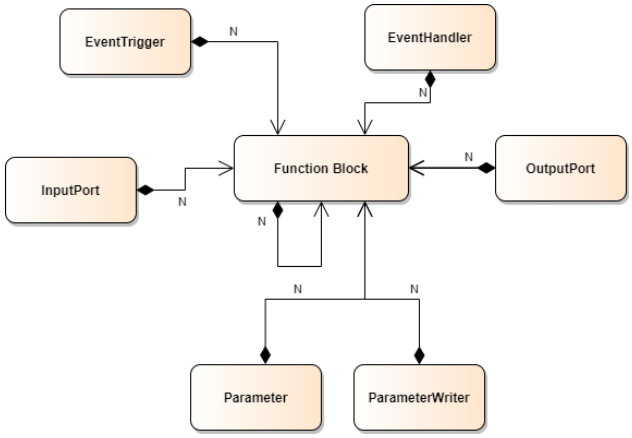


Figure 3: Schematic of a function block.

Processing of Function Blocks in a Thread

FBs are instantiated and serialized in the loading phase of the RTF context. Thereby, multiple function blocks execute as a chain on the thread where they are deployed. The ordering of function blocks is determined by the relation between function blocks once the configuration is parsed.

Synchronization in real-time threads executing the FBs is implemented with busy-wait rather than through interrupts or callbacks to avoid context switching and effectively minimize jitter and response times. Thus, RT application runs properly once it has an appropriate CPU allocation, schedule policy and priority attributes.

On start (or re-entry) of the thread, all FBs are reset. Then, the thread enters the active state, which has the following stages (Fig. 4).

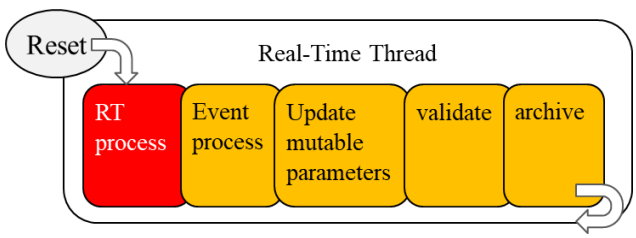


Figure 4: Processing of Function Block in a Thread.

- **RT process**: handles processing of the logic. All the inputs/outputs get updated in this stage.
- **Process events**: handles processing of all the events that have been triggered.
- **Process parameters**: updates the mutable parameters.
- **Validate**: validates the execution times and/or parameters to be validated.
- **Archive**: archives signals and all archivable objects.

The RT process is ideal for a hard real-time mode operation and should be constant over all the cycles. The underneath rule of execution of FB is to execute process method periodically under rt-thread, and thus ensure predictability of execution times. Fully data-driven mechanism requires the FB processing logic to be implemented in a non-blocking way. The remaining four consecutive jobs are executed in the offline real-time mode.

Framework Life Cycle Management

Since the framework is a part of a distributed control system, multiple instances must be centrally controlled in an organized manner. The Life Cycle Management Service (LCMS) allows to centrally orchestrate the state transition along with loading of the configuration.

In order to increase performance and reliability, it is desired to maintain the core real-time engine of RTF as lightweight as possible, and handle all configuration loading, parsing and expansion logic in an external system/process. Portability is also ensured by exposing LCMS interfaces through an API which allows site-specific protocol implementations. Two protocols are currently supported; pvAccess protocol from EPICS v7 and native TCP/IP.

Although EPICS has been selected as a standard middleware for ITER control, the evolution of configuration propagation and the need for improvement of code in terms of the software QA process have been raised. RTF renovated the primarily interface function based on the PVXS [6] as thereby.

Figure 5 shows limited number of PVs that are created just after instantiation of the RTF and are ideal for operational control. Fig. 6 depicts RTF state machine operation addressed through the {}-RTF-OPSTATE PV. After loading the configuration, additional application specific PVs will be dynamically created and published. In order to support client in a conventional manner, RTF uses the normative type, which creates a structure holding the value, alarm, and time structure.

```
leew2 @ diag-fc1.codac.iter.org : ~ $ pvlist localhost
CTRL_PCS:N1-LOAD-APP
CTRL_PCS:N1-LOAD-SERVICE
CTRL_PCS:N1-RTF-OPREQ
CTRL_PCS:N1-RTF-OPSTATE
CTRL_PCS:N1-RTF-RESET
[ 12:26:17 ]
leew2 @ diag-fc1.codac.iter.org : ~ $
```

Figure 5: Default PVs shown by EPICS bundle Command Line Interface (CLI) tool, which are minimized for configuration transmission and operation after RTF instantiation.

CONFIGURATION AND DEPLOYMENT

The RTF provides four types of configurations in an XML-based format.

- **Environment configuration** defines the computer nodes (hosts), processes and threads that host the RTF services and RT applications.
- **Service configuration** defines the services of the RTF that provide infrastructural capabilities such as a console, logging, archiving and communications between processes and across hosts.
- **RT application configuration** defines the RT applications in terms of configurable interlinked FBs that take inputs, perform processing and produce outputs.
- **Deployment configuration** assigns the services and RT applications to threads defined by the environment.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

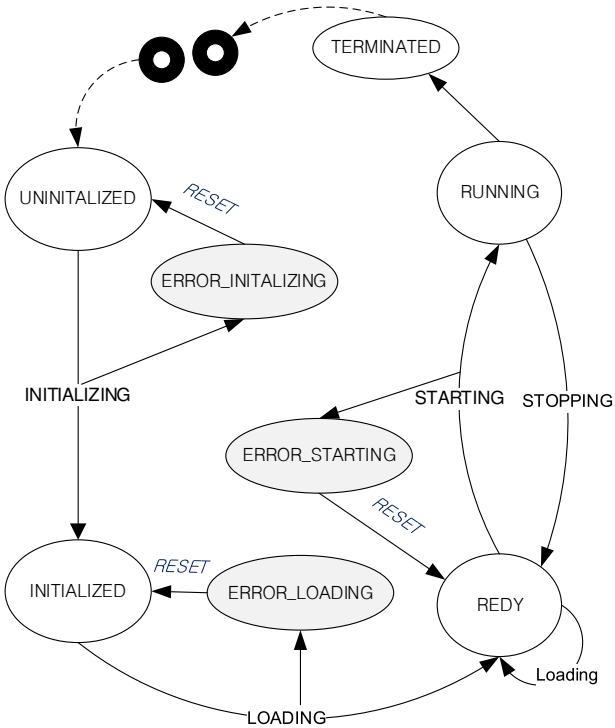


Figure 6: State diagram controlled by LCMS. *{PRFIX}-RTF-OPSTATE* PV indicates RTF state transition.

For the pvAccess based LCMS, the environment and service configuration is transmitted via *"{PREFIX}-LOAD-SERVICE"*, and *"{PREFIX}-LOAD-APP"* is assigned for the application and deployment configuration. Since the application configuration is decoupled from the deployment configuration, the RT application can be run on different computing resource defined in the environment configuration without any changes to its configuration or code even in pulse-to-pulse operation, while the environment and service configuration likely unnecessary to change.

Configuration Parser

The Configuration parser parses the input XML configuration files and inserts any necessary logic to create a fully expanded configuration. In general, parser deals with whole configuration, then each RTF process extracts those parts of configuration which they are supposed to run. This fully expanded configuration is then converted to an in-memory representation of the configuration which is used by the real-time process. The parser is decoupled from the framework. The RTF provides C++ API for facilitating it into other programs as standalone console executable, *rtftool*, or other supervision application, Supervision (SUP) in ITER.

INTERFACE SUPPORT

Great emphasis has been put into the RTF to fully integrate it into the CODAC and to support all main interfaces such as Synchronous Databus Network (SDN), Data Archiving Network (DAN), and EPICS pvAccess protocol. The Nominal Device Support (NDS) service which is an intermediate layer for the physical hardware interfacing is under development. Foundational networking infrastructure is implemented as a transport layer service and therefore supports communication on all levels of RTF.

The RTF also supports Simulink interface based on automatic code generation function. A direct transfer of a Simulink model to a real-time framework FB can be achieved and successfully executed. Fig. 7 shows how compiled library from Simulink can be used inside RTF through the application configuration. This offers an extremely powerful method to obtain the necessary code directly from a model.

```
<FunctionBlock Name="KCURR" Type="SimulinkBlock">
  <Parameter Name="LibraryPath" Value="build/libKCURRsim.so"/>
  <Parameter Name="BlockName" Value="KCURRsim"/>
  <Parameter Name="Ky" Value="[[0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,
  <Parameter Name="Nd" Value="100"/>
  <Parameter Name="P" Value="[1,1,1,1,1,1,1,1,1,1,1]"/>
  <Parameter Name="D" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="I" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="Vlow" Value="[-1050,-1050,-2100,-1050,-1050,
  <Parameter Name="Vup" Value="[1050,1050,2100,1050,1050,1050,3

  <InputPort Name="V_FF" Signal="loaderBuffer1"/>
  <InputPort Name="I_CSPF_ref" Signal="loaderBuffer2"/>
  <InputPort Name="I_CSPF" Signal="loaderBuffer3"/>
  <InputPort Name="FlagInom" Signal="loaderLoader4"/>
  <InputPort Name="enable" Signal="loaderLoader5"/>

  <OutputPort Name="u" Signal="OutputSIM1"/>
  <OutputPort Name="PIDinternalSignals" Signal="OutputSIM2"/>
  <OutputPort Name="errorSignal" Signal="OutputSIM3"/>
</FunctionBlock>
```

Figure 7: Example how Simulink wrapper FB is configured in the application configuration, which loads the compiled library. Non-scalar data types are also fully supported.

CASE STUDY USING RTF

Multiple use case studies are being conducted from the IO and fusion community in order to identify possible limitations and improvements in parallel with implementation of RTF.

PCS Prototyping

Prototyping is an efficient way to cross-check the detailed intention of design and its implementation. The PCS components are sufficiently modularized to support as other systems become available, however the architecture remain constant for ITER lifetime. Thus it ideal for RTF capturing specific function individually and integrated efficiently along with project progress.

The PCS Compact Controller (CC) is a continuous controller among architectural components for versatile control purpose. It is representative as a Proportional, Integral, and Derivative (PID) controller with fine-tuned attributes tailored for plasma parameter control. As first work, CODAC group implemented 19 FBs including trajectory block and plant model for the simulation. The FBs are nearly map to the individual Simulink block, verified fidelity of outputs between Simulink and RTF traces [7].

Since the PCS design development is based on components running within the Simulink platform, it is under consideration the possibility to convert components from Simulink model into components compatible with the RTF in a quasi-automatic manner [8].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL02

Consequently, Simulink wrapper function had been devised and verified proper transposition from model to application with high fidelity of result. Customised FB, *"SimulinkBlock"*, loads the compiled library, which was automatically built by the RTF CLI.

Underlying concept is supporting the mutable parameter in the same manner as conventional FB. Therefore, model designer configures the desired function only by changing parameters, while maintaining the same external interface to the other FB. Even if model changed, thereby recompile the generated code, RTF FB always persists independently.

Plasma Diagnostics Data Processing

Edge Thomson Scattering (ETS) diagnostics has been demonstrated in the running tokamak. The Thomson Scattering diagnostics gives reliable electron temperature and density profiles in magnetically confined plasma. A 5GS/s CAEN DT5742 digitizer [9] operates in pulse mode synchronized with Nd:YAG Laser system where has up to 50Hz injection rate [10]. The customized data acquisition FB archives raw data through RTF transport layer whilst the output links to the fitting FB to eliminate back scattered signal. Passed series of signal conditioning, electron temperature is measured using lookup table where calibrated data is stored as per the wavelength from the polychromator signal.

The developed prototype covering complete data acquisition, processing path, archiving as well as measurements publishing can be used as a reference example for other ITER diagnostic systems.

Poloidal Field Coil Control for Plasma Start-up

The most fundamental control module of PCS is coil power supply for discharge control. Poloidal Field (PF) coil is a main actuator for plasma breakdown and thereafter shape and position control. CODAC commenced implementing a real-life controller in order to evaluate both functional and non-functional behaviour of the PCS with collaboration of Korea Superconducting Tokamak Advanced Research (KSTAR) control team.

The full-featured mini-CODAC provides all ITER standard networking protocols such as real-time data communication, experimental data archiver and time synchronization. Additional installation of the RTF along with the PCS platform library facilitates building controller following the PCS architecture design.

11 PF controllers were devised complying with KSTAR native function model. Minimum protection function took into accounted, verified 20kHz run cycle in consecutive process pipeline such as exception handler, waveform generator, and PID function. Integrated operability was verified by implementing site-dependent interface functions such as for MDSplus, Reflective Memory network, and EPICS CA.

CONCLUSION

The RTF is a flexible high-performance software platform that facilitates the development and deployment of complex real-time applications. It was designed to be portable and modular, enabling high reusability and maintainability of components constituting the real-time applications.

Factory design pattern, and rich function for multi-threaded program enables building application through configuration-driven process.

Prototyping activities on some of the operating Tokamaks have demonstrated its applicability for the implementation of ITER real-time control systems.

ACKNOWLEDGEMENTS

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

[1] J.A. Snipes *et al.*, "ITER Plasma Control System Final Design and Preparation for First Plasma", *Nuclear Fusion,* vol. 61, no. 10, Sept. 2021. doi:10.1088/1741-4326/ac2339

[2] A. Winter *et al.,* "The ITER real-time framework final design", presented at the 11th IAEA Technical Meeting on CODAC, Greifswald, Germany, May 2017.

[3] W. Lee *et al.,* "Software Architecture and Design Document for the ITER Real-time Framework", ITER Internal Communication.

[4] Simulink, https://www.mathworks.com/help/simulink

[5] NI LabView, https://www.ni.com/labview.html

[6] https://mdavidsaver.github.io/pvxs/index.html

[7] L. Zabeo, B. Bauvir, W. Lee, *et al.*, "PCS Proto-typing activity", ITER internal communication.

[8] L. Zabeo *et al.*, "Work-flow process from simulation to operation for the Plasma Control System for the ITER first plasma", *Fusion Eng. Des.,* vol .146, pp. 1146-1149, 2019. doi:10.1016/j.fusengdes.2019.02.101

[9] https://www.caen.it/products/dt5742/

[10] J.H. Lee *et al.*, "Research of Fast DAQ system in KSTAR Thomson scattering diagnostic", presented at the 18th Laser Aided Plasma Diagnostics (LAPD18), Prague, Sep. 2017. doi:10.1088/1748-0221/12/12/C12035

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

MACHINE LEARNING PLATFORM: DEPLOYING AND MANAGING MODELS IN THE CERN CONTROL SYSTEM

J.-B. de Martel, R. Gorbonosov, Dr. N. Madysa, CERN, Geneva, Switzerland

Abstract

Recent advances make machine learning (ML) a powerful tool to cope with the inherent complexity of accelerators, the large number of degrees of freedom and continuously drifting machine characteristics.

A diverse set of ML ecosystems, frameworks and tools are already being used at CERN for a variety of use cases such as optimization, anomaly detection and forecasting. We have adopted a unified approach to model storage, versioning and deployment which accommodates this diversity, and we apply software engineering best practices to achieve the reproducibility needed in the mission-critical context of particle accelerator controls.

This paper describes CERN Machine Learning Platform (MLP) - our central platform for storing, versioning and deploying ML models in the CERN Control Center. We present a unified solution which allows users to create, update and deploy models with minimal effort, without constraining their workflow or restricting their choice of tools. It also provides tooling to automate seamless model updates as the machine characteristics evolve. Moreover, the system allows model developers to focus on domain-specific development by abstracting infrastructural concerns.

MOTIVATION

Machine learning techniques and in particular neural networks are well suited to the unique challenges of particle accelerator controls [1]. Neural networks are already being used in CERN controls for a variety of use cases including anomaly detection [2], trajectory steering at LINAC4 and AWAKE [3], beam measurements [4] and collimator alignment [5] in the LHC.

In recent years, the rapid expansion of the ML ecosystem and the emergence of MLOps has created a multitude of tools and frameworks to assist data scientists with different aspects of the ML development workflow. These include tooling for experiment tracking and model management (e.g. Neptune [6], Comet [7]), feature storage (e.g. Feast [8]), pipeline and workflow automation (e.g. Pachyderm [9], Airflow [10]), hyper-parameter tuning (e.g. Katib [11], Sigopt [12]), deployment (e.g. Seldon [13]) and monitoring (e.g. Fiddler [14], Evidently [15]). Comprehensive tools which aim to address the whole ML lifecycle also exist, both open source (e.g. MLFlow [16], Kubeflow [17]) and proprietary (e.g. AWS Sagemaker [18], GCP Vertex AI [19]).

However, none of these comprehensive tools fit the use-cases required by CERN controls – they either constrain model developers' workflows or require in-depth knowledge of infrastructural tooling. Furthermore, these tools do not fully address requirements specific to accelerator controls such as high criticality, continuously drifting machine characteristics, variety of use-cases (online and offline, embedded and standalone) and the need to maintain different model configurations for each accelerator beam type.

For these reasons we present a machine learning platform (MLP) specific to CERN controls. It addresses the aforementioned issues by abstracting and simplifying model management, storage, and deployment concerns. In addition, it is open and extensible by design to cope with the rapidly evolving ML landscape and lack of generally accepted industry standard for MLOps. For the same reason, it is designed to be compatible with diverse ML model training environments (local, CERN infrastructure, and public cloud). Helping with rapid development of new models with tools such as experiment tracking or workflow automation are not goals of MLP – instead, it is designed to integrate with existing solutions.

CONCEPTS

We define models as the combination of a model type and model parameters. Model types contain the algorithm and logic of the model, e.g., the neural network architecture, the framework, and data pre- and post-processing. Model parameters are the data which configures the model type, e.g. trained weights of neural networks, and any other configuration variables. As the format of model parameters is highly dependent on the framework used[1], we decided to treat model parameters as opaque data, which we store but don't inspect within MLP.

A given model type can be associated with multiple model parameters. One use case for this is the use of different model parameters for each type of particle beam produced by the accelerators. The opposite is also true, given model parameters can be associated with different model types. For example, a given set of trained neural network weights can be used by a same model surrounded by different pre- and post- processing logic for different use cases.

Model types and model parameters evolve independently and are versioned separately, so we define model type versions (MTV) and model parameters versions (MPV). MTVs and MPVs compatibility follows a many-to-many relationship, as shown in Fig. 1.

The combination of an MTV and a compatible MPV forms a model. Models are fully configured neural networks or

[1] Common formats such as ONNX [20] exist but don't support certain operations such as custom layers or loss functions.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

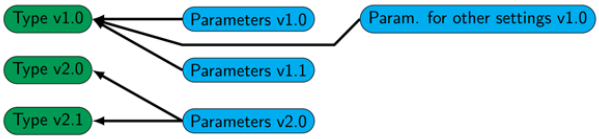


Figure 1: Model types, models parameters and compatibility.

other algorithms which can perform predictions. They expose a uniform prediction API. Models can be both embedded into a client process or deployed as a standalone process and accessed remotely.

MODEL DEVELOPMENT AND USAGE WORKFLOW

MLP has been designed in collaboration with ML model developers and follows their general workflow without restricting it. This section describes MLP features in the order in which they appear in a typical workflow.

Defining a Model

Developers define models exposing a common interface, the MLP Model API. This interface defines four operations:

1. a *fit* operation to train the model on the provided input data;
2. an *export parameters* operation to extract the current values of all model parameters;
3. an *import parameters* operation to configure the model using the provided parameters;
4. and a *predict* operation to return a prediction from the input data.

Registering Model Types

MLP aims to minimize the impact on model developers' workflow by automating management actions as much as possible. Nowadays the majority of model developers use Git as a version control system for their source code, and often use git tags to label particular versions of interest. MLP leverages git tags to automate model type registration: model types are registered automatically when developers push new tagged versions of their model code. A single project can contain multiple models (for example, this is crucial for composite models such as Auto-Encoders [21]). When a new version of the project code is pushed, one MTV is automatically registered within MLP for each model belonging to the project.

Publishing Model Parameters

Model parameters registration cannot re-use the same mechanism as model types since model parameters are created programmatically, practically never stored in git due to their large size and usually created at the end of a lengthy training process. Without using MLP, many model developers at CERN export the trained parameters to disk or an external service within the training script. This requires handling infrastructural concerns such as where to store these weights, how to version them, how to share them with other developers, monitoring the disk space available, etc. This often results in inconsistent naming patterns and file locations which creates additional complexity and a maintenance challenge.

MLP addresses these concerns by providing a client library to register trained parameters. Developers simply provide the model instance, a name for the trained parameters and a version number.

In continuous retraining use-cases, users can also leverage the version generation feature and let MLP decide which version number to assign to the new model parameters version to avoid implementing a complex versioning algorithm themselves.

Based on semantic versioning [22], the automated versioning logic is optimized for common use cases and looks at a limited amount of information to make an informed decision: the currently used MTV and the compatible MPVs registered in MLP. It will refuse to guess in the face of ambiguity. Table 1 illustrates the version guessing behavior for the most common cases.

Table 1: Model Parameters Version Number Generation

MTV	Highest MPV	⟹	Generated MPV
1.0.0	none exist yet	⟹	1.0
1.0.0	1.0	⟹	1.1
1.6.0	1.1	⟹	1.2
2.0.0	1.2	⟹	2.0
3.3.0	4.0 (no 3.x)	⟹	ambiguity
3.3.0	4.0 (3.3 exists)	⟹	3.4

Managing Compatibility

The default behavior of the platform is optimized for the common use case while leaving the model developer in full control. When a MPV is published from a MTV instance, a compatibility link between them is created automatically.

Furthermore, MLP, based on semantic versioning, makes new MTVs inherit all the compatible MPVs from the previous MTV with the same major version number. The same applies to MPVs. When a breaking change is made to a model type or model parameters and compatibilities should not be copied from the previous version, model developers should bump the major version to indicate the backward-incompatibility to MLP and then the compatibilities will not be copied. For niche use cases, it is also possible to disable compatibility copying on creation. Compatibility links can also be added and removed manually using the client library.

Using Embedded Models

To embed an MLP model in an application written in the same language as the model, developers should use the client library to instantiate an embedded model. The client

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

application must specify the class of the locally available model type version and provide a parameters name and version. The client library will then take care of retrieving the appropriate MPV and load them into the model.

Users can also choose to always use the latest compatible parameters version. This is the right choice for most use cases. Together with automatic versioning, this provides a way to update running models automatically.

Using Standalone Models

MLP models can also be used as standalone models and accessed remotely using the standalone model API. Users connect to the model by specifying the model type name and model parameters name; they can then use the model as if it was available locally.

This not only allows models to be called from any language that supports HTTP requests, but also enables client applications to always use the latest models with no effort required from application developers. Standalone models serve prediction requests over the network, allowing them to be called from any language, not just the language in which they were written.

IMPLEMENTATION

The main components of the implementation are the model API, the client library, the model registry server, and the standalone serving cluster. A simplified overview of the system architecture is presented in Fig. 2.

Model API

Today, the machine learning landscape is heavily dominated by Python and virtually all models at CERN are implemented in the Python language. For this reason, MLP models are currently restricted to Python and the model API is implemented using Python abstract base classes [23]. A model class must implement the *MlpModel* interface (see Fig. 3) to be considered a model type by the Machine Learning Platform. This is usually accomplished through direct inheritance, although other methods for niche use cases exist. Default extensible implementations of parameter saving and loading logic are provided for commonly used frameworks (tensorflow [24], pytorch [25], scikit-learn [26]) to facilitate the implementation.

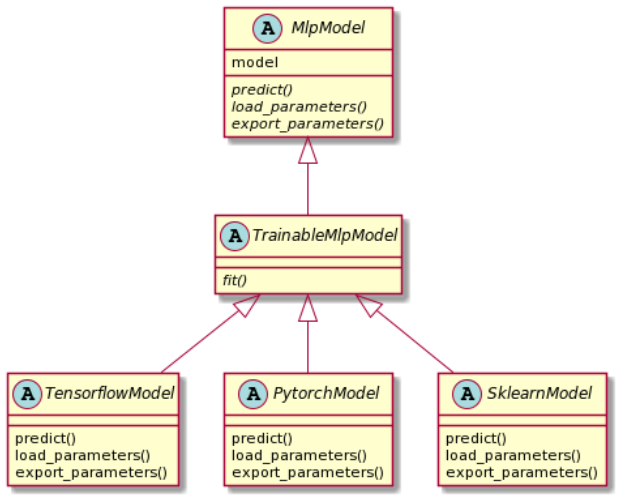


Figure 3: Class Diagram of the MLP Model API.

Model Registry Server

The model registry server is responsible for storing and managing MTVs, MPVs and their compatibilities. It also provides the version generation logic described above. It is implemented as a standard Java/Spring Boot application exposing REST endpoints and documented with Swagger. It uses a relational database to store MTV, MPV and compatibility metadata and an object storage service to store MPV binary objects (trained weights). The relational database is an Oracle database versioned with Liquibase [27]. The object storage service currently uses a CERN NFS service but will change to the Openstack Object Store (Swift) in the near future.

Client Library

MLP provides a single client library for model developers and users alike. Distributed as a Python package, it allows

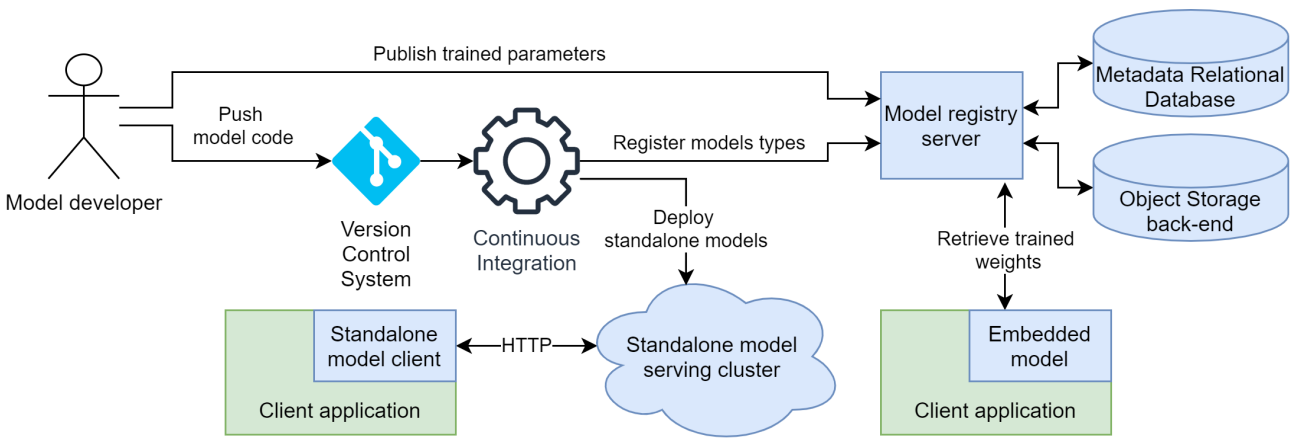


Figure 2: Simplified architecture diagram. Highlighted in blue are the components provided by MLP.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

model users to publish model parameters, perform search queries against the model registry, and instantiate and use both embedded (see Listing 2) and standalone (see Listing 5) models.

To publish the model parameters of a trained model to MLP, model developers instantiate the MLP client and call a publishing method, providing the trained model, a name for the parameters, and an optional version number, as shown in Listing 1.

```
from mlp_client import Client, Profile, AUTO
from my_model import MyModel

model = BeamLineModel()
model.fit(...)

client = Client(Profile.PRO)
client.publish_model_parameters_version(
    model,
    name="proton_beam_config",
    version=AUTO, # generated by server
)
```

Listing 1: Publishing model parameters versions

When re-training a model continuously, the combination of automatic version generation in the training process (Listing 1) and automatic version selection in the consumer process (Listing 2) makes it easy to set up continuous retraining processes without implementing complex version management logic on both the training and consumer sides.

Registration Automation

Automatic model type registration is accomplished using Continuous Integration (CI). Model developers only need to include a Gitlab CI template provided by MLP. As a single project can contain multiple models, model developers must also register their models as Python entry points under a specific key, as shown in Listing 3.

When model developers push a git tag, the CI jobs defined by the template will publish the package containing the model code to a central repository. It will then iterate over all the models declared in the package and publish them to the model registry server. The version of the new model types is determined from the name of the tag. The provided CI template can be extended with custom steps, such as linting and testing jobs, as shown in Listing 4, leaving model developers complete control of their CI pipelines and allowing them to add additional model validation steps if appropriate.

```
from mlp_client import Client, Profile, AUTO
from my_model import MyModel

client = Client(Profile.PRO)
model = client.create_model(
    model_class=BeamLineModel,
    params_name="ion_beam_config",
    params_version=AUTO
)

result = model.predict(data)
```

Listing 2: Using embedded models

```
[options.entry_points]
mlp_models =
    model_1 = awake:VaeEncoder
    model_2 = awake:VaeDecoder
```

Listing 3: Model type declaration in setup.cfg

```
include:
- project: machine-learning-platform/mlp-ci
  file:mlp-ci-template.yml

variables:
 project_name: simple_ann

my_custom_test_job:
 script: ./custom-test-script.py
```

Listing 4: Including and extending the Gitlab CI template

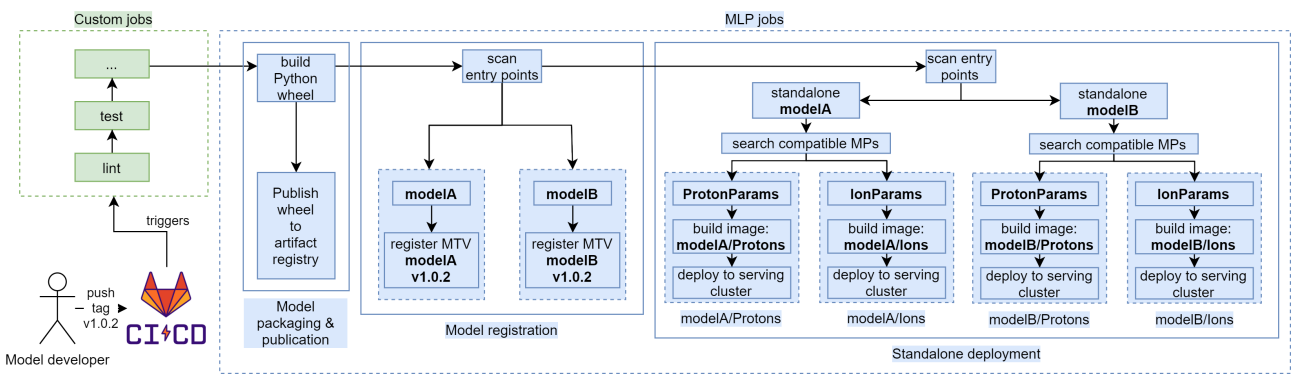


Figure 4: Simplified CI diagram.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Standalone Deployment

If the model is declared as standalone, a standalone deployment job is also run. For each model type declared in the model developer's project, the standalone deployment job will identify all compatible model type / model parameters combinations and deploy the latest version of each pair to the standalone serving cluster. To deploy the models to the serving cluster, the standalone deployment job first builds container images which include the MTV, the MPV and a light Python web server. The produced images are deployed to the standalone serving cluster in a subsequent CI job, replacing outdated models if applicable. A simplified diagram of the pipeline is shown in Fig. 4. When the container is started, the web server listens for prediction requests, forwards them to the MTV/MPV combination, and returns a serialized result to the remote caller.

Standalone Serving

The standalone serving component, still in the prototype phase [28], is responsible for serving model predictions to remote clients over HTTP. It is currently implemented as a Kubernetes cluster with an Nginx ingress which listens for prediction requests and forwards them to the appropriate model container. Users can access standalone models using the Python client and use them in a similar way to embedded models, as shown in Listing 5. At the moment, a simple json-based serialization mechanism is used and will be improved in the future.

```
from mlp_client import Client, Profile, AUTO
from my_model import MyModel

client = Client(Profile.PRO)
model = client.create_standalone_model(
    model_class="BeamLineModel",
    params_name="proton_beam_config",
    params_version=AUTO
)
result = model.predict(data)
```

Listing 5: Using standalone models

Technical solutions to autoscale model containers based on usage are currently under investigation. Candidates include the Kubernetes Horizontal Pod Autoscaler [29], KEDA [30] and KFServing [31].

ACHIEVEMENTS AND FUTURE PLANS

Despite its early prototype status, MLP is already undergoing user testing for an application in the AWAKE experiment [32] using embedded models. It is also being evaluated for prediction tasks at the Super Proton Synchrotron, CERN's second largest accelerator, using convolutional neural networks.

The future plans of MLP are primarily focused on integration with other systems, including ML-specific tools and CERN internal services. ML-specific tools include experiment management tools such as Neptune [6] or Weights & Biases [33] and workflow and pipeline automation tools such as Apache Airflow [10]. CERN internal tools include a generic GUI application for numeric optimization, the accelerator time-series data logging service (NXCALS [34]), and settings management tools (INCA/LSA [35]).

Other plans include opening MLP to public cloud services to simplify model training on external infrastructure and adding user-defined metadata to models to enhance search capabilities.

REFERENCES

[1] A. L. Edelen, S. G. Biedron, B. E. Chase, D. Edstrom, S. V. Milton, and P. Stabile, "Neural networks for modeling and control of particle accelerators," *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 878–897, 2016.

[2] F. Tilaro, B. Bradu, M. Gonzalez-Berges, M. Roshchin, and F. Varela, "Model learning algorithms for anomaly detection in CERN control systems," in *16th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2018, p. TUCPA04.

[3] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Z. Della Porta, N. Bruchon, and G. Valentino, "Sample-efficient reinforcement learning for cern accelerator control," *Phys. Rev. Accel. Beams*, vol. 23, p. 124801, Dec 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevAccelBeams.23.124801

[4] P. Arpaia, G. Azzopardi, F. Blanc, X. Buffat, L. Coyle, E. Fol, F. Giordano, M. Giovannozzi, T. Pieloni, R. Prevete, S. Redaelli, B. Salvachua, B. Salvant, M. Schenk, M. S. Camillocci, R. Tomás, G. Valentino, F. F. V. der Veken, and J. Wenninger, "Beam measurements and machine learning at the cern large hadron collider," 2021.

[5] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Operational results of LHC collimator alignment using machine learning," in *10th International Particle Accelerator Conference*, 2019, p. TUZZPLM1.

[6] "Neptune: experiment management and collaboration tool," https://neptune.ai.

[7] "Comet," https://www.comet.ml/.

[8] "Feast.dev," https://feast.dev/.

[9] "Pachyderm," https://www.pachyderm.com/.

[10] Apache Software Foundation, "Apache Airflow," https://airflow.apache.org/.

[11] J. George, C. Gao, R. Liu, H. G. Liu, Y. Tang, R. Pydipaty, and A. K. Saha, "A scalable and cloud-native hyperparameter tuning system," 2020.

[12] S. Clark and P. Hayes, "SigOpt," https://sigopt.com, 2019.

[13] C. Cox, G. Sunner, A. Saucedo, R. Dawson, A. Gonzalez, and R. Skolasinski, "Seldon Core: A framework to deploy, manage and scale your production machine learning to thousands of models," https://github.com/SeldonIO/seldon-core, 2018.

[14] "Fiddler," https://www.fiddler.ai.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL03

[15] "Evidently AI," https://evidentlyai.com/.

[16] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, A. Singh, F. Xie, M. Zaharia, R. Zang, J. Zheng, and C. Zumar, "Developments in mlflow: A system to accelerate the machine learning lifecycle," *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, 2020.

[17] "Kubeflow," https://www.kubeflow.org/.

[18] "AWS Sagemaker," https://aws.amazon.com/sagemaker/.

[19] "GCP Vertex AI," https://cloud.google.com/vertex-ai.

[20] J. Bai, F. Lu, K. Zhang *et al.*, "ONNX: Open Neural Network Exchange," https://github.com/onnx/onnx, 2019.

[21] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *arXiv e-prints*, p. arXiv:1906.02691, Jun. 2019.

[22] "Semantic versioning 2.0.0," https://semver.org.

[23] "Abstract Base Classes," https://docs.python.org/3/library/abc.html.

[24] "Tensorflow," https://www.tensorflow.org.

[25] "Pytorch," https://pytorch.org.

[26] "Scikit-learn," https://scikit-learn.org/stable.

[27] "Liquibase," https://www.liquibase.org.

[28] R. Voirin, T. Oulevey, and M. Vanden Eynden, "The State of Containerization in CERN Accelerator Controls," in *18th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'21), Shanghai, China*, 2021.

[29] "Kubernetes Horizontal Pod Autoscaler," https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale.

[30] "Kubernetes Event-Driven Autoscaling," https://keda.sh.

[31] "KFServing," https://github.com/kubeflow/kfserving.

[32] E. Adli, A. Ahuja, O. Apsimon, R. Apsimon, A.-M. Bachmann, D. Barrientos, F. Batsch, J. Bauche, V. B. Olsen, M. Bernardini *et al.*, "Acceleration of electrons in the plasma wakefield of a proton bunch," *Nature*, vol. 561, no. 7723, pp. 363–367, 2018.

[33] "Weights and Biases," https://wandb.ai.

[34] J. Wozniak, C. Roderick, and S. R. WEPHA163, "Nxcals-architecture and challenges of the next cern accelerator logging service," in *17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA*, 2019.

[35] D. Jacquet, R. Gorbonosov, and G. Kruk, "LSA - the High Level Application Software of the LHC - and Its Performance During the First Three Years of Operation," in *14th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'13), San Francisco, CA, USA*, 2013.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

KARABO DATA LOGGING: InfluxDB BACKEND AND GRAFANA UI

G. Flucke*, V. Bondar, R. Costa, W. Ehsan, S. G. Esenov, R. Fabbri, G. Giovanetti, D. Goeries, S. Hauf, D. G. Hickin, A. Klimovskaia, A. Lein, L. Maia, D. Mamchyk, A. Parenti, G. Previtali, A. Silenzi, D. P. Spruce[1], J. Szuba, M. Teichmann, K. Wrona, C. Youngman
European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany
[1]*now at* MAX IV, Fotongatan 2, 22484 Lund, Sweden

Abstract

The photon beam lines and instruments at the European XFEL (EuXFEL) are operated using the Karabo control system that has been developed in house since 2011. Monitoring and incident analysis requires quick access to historic values of control data. While Karabo's original custom-built text-file-based data logging system suits well for small systems, a time series data base offers in general a faster data access, as well as advanced data filtering, aggregation and reduction options. EuXFEL has chosen InfluxDB as backend that is operated since summer 2020. Historic data can be displayed as before via the Karabo GUI or now also via the powerful Grafana web interface. The latter is e.g. used heavily in the new Data Operation Center of the EuXFEL. This contribution describes the InfluxDB setup, its transparent integration into Karabo and the experiences gained since it is in operation.

KARABO AND THE EuXFEL

The European X-ray Free Electron Laser (EuXFEL) facility [1] provides hard and soft X-ray beams at three photon beamlines to six instruments. Up to 27,000 photon pulses per second are arranged into 10 Hz trains with an intra-train pulse repetition rate of 4.5 MHz. The Karabo framework [2–4] has been designed and developed in-house since 2011 for control, online data analysis, and data acquisition at the photon beam lines and the scientific instruments.

In Karabo, so-called devices communicate via a central message broker. All devices using the same broker *topic* form a Karabo installation. Whereas broker communication is considered to be "slow" data, big or "fast" data like images are sent via TCP/IP data pipelines that can be flexibly configured, e.g. for calibration, analysis, or preview purposes. Figure 1 gives an overview of a Karabo installation.

A Karabo device exposes a self-description of its control interface, i.e. its *schema*. Karabo's generic graphical user interface (GUI) uses the schema to render the representation of a device. Devices can have one of manifold tasks:

- interface some hardware like a pump or a motor,
- control a detector and read out its data,
- analyse data,
- orchestrate other devices,

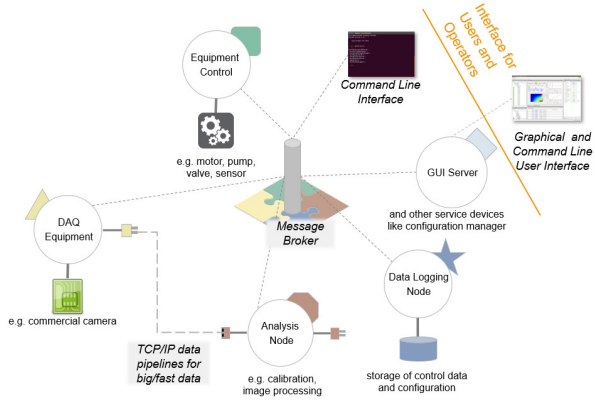


Figure 1: A Karabo installation showing Karabo devices with various tasks. Broker and pipeline communication lines are indicated.

- provide a system service like serving as entry point for the GUI, logging data, managing alarm states, or managing configurations.

To communicate with each other, the Karabo devices expose methods that can be called remotely in the distributed system. Besides being directly called, these *slots* can be subscribed to *signals* of other devices. When such a signal is emitted with arguments, all subscribed slots are called with these arguments. In the process, only a single message is sent to the broker that distributes the message according to the subscriptions, as is shown in Fig. 2. This signal/slot mechanism allows Karabo to be fully event-driven, regular polling of e.g. device properties is not needed. That a single message to the broker is sufficient also for a device with a signal that many other devices have subscribed to, ensures that there is no overhead for such a "popular" device.

FIRST KARABO DATA LOGGING IMPLEMENTATION

Data logging in a Karabo installation is organised via a few dedicated devices. A "data logger" device (or several that share the load) subscribes to the signal for property updates of the other devices. Properties are configuration parameters or read-only values like the reading of a temperature sensor. Via the signal/slot mechanism, the logger is informed about every property update and when this update occurred, i.e. the timestamp of the update, and stores it in the backend of the logger. Similarly, the device schema and its potential

* gero.flucke@xfel.eu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

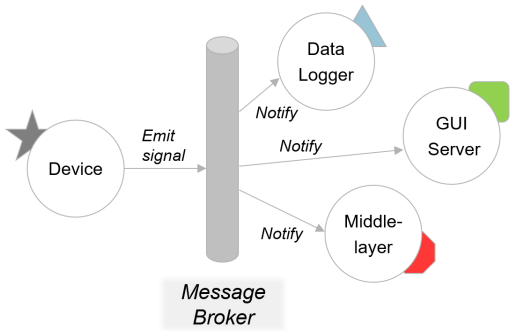


Figure 2: Karabo signal/slot messaging via the broker.

run-time updates are stored. "Data log reader" devices are responsible to read back the stored data. One can query either the changes of a property in a time interval (*trendline* data) or all properties of a device and its schema at a given point in time. A single "Data logger manager" device per Karabo installation orchestrates loggers and readers.

The first Karabo data logging implementation, started in 2014, was based on human readable ascii files. Each file logs properties of a single device with one line per property:

```
20200719T184128.573059Z
  |1595184088.573059|804227972
  |actualPosition|FLOAT|35.91043||VALID
```

Besides the timestamp in two formats and the corresponding id of the EuXFEL train of photon bunches, the name, type and value of the property, and some meta information is logged. A further file stores the device schema, serialised to an XML format. To reasonably speed-up reading all property updates in a time interval, binary index files are created per property of interest. Figure 3 shows the result of such trendline requests in the Karabo GUI. The actual position of a motor and its state are displayed, as part of an incident analysis. Since Karabo is event-driven, no new data points exist when the motor is not moving. The zoom into the end of the curve shows that the motor decelerated and got stuck – in fact, the detector moved by the motor collided with another object since, after a change of the setup, the limit switch that would prevent such collisions was unfortunately not adjusted.

Due to limited screen resolution, a GUI cannot display arbitrary details of a trendline. Therefore, the trendline data request specifies a maximum number of points to return. If the requested time interval contains more data points, a simple down-sampling algorithm is applied: just every 2nd, 3rd, etc. data point is returned.

Drawbacks

Deployment of such a logging and data retrieval mechanism is easy since it depends only on the availability of disc space and therefore suits well for small Karabo installations. On the other hand, operation at EuXFEL reveals some (obvious) drawbacks. Storing data in text files does not scale well concerning disc space. At EuXFEL, data older than three months were automatically moved to an archival

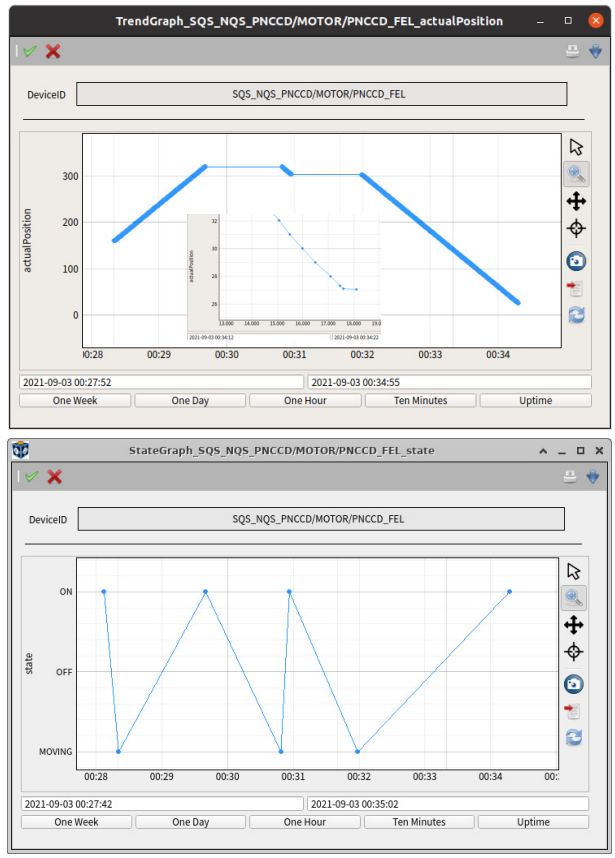


Figure 3: Historic trendline data for the position in mm of a motor that is moved back and forth, overlayed with a zoom into the right end of the curve (top), and the corresponding motor state changes (bottom).

location. Hence, access to such data required extra effort. Furthermore, data read-out could be slow: depending on the number of property updates since a device was last restarted, reading back the configuration of a device at a given point in time could take in the order of minutes. The lack of statistical methods for data downsampling could result in the loss of short timescale features in the data. Two requests of the same data with only slightly differing time intervals could return markedly different data, due to the sparsing algorithm that was implemented.

TRANSITION TO AN INFLUXDB BACKEND

Given the drawbacks of the ascii file based solution, the Karabo development team looked for a better storage backend. Timeseries databases were considered to be particularly well suited, as they are optimised for retrieving data along the time axis. InfluxDB [5], Prometheus [6], and Timescale [7] were considered and finally InfluxDB was chosen. A prototype Karabo device interacting with an InfluxDB proved the feasibility in 2018.

In order to allow for a transparent transition and to keep the text based logging backend available for small Karabo

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

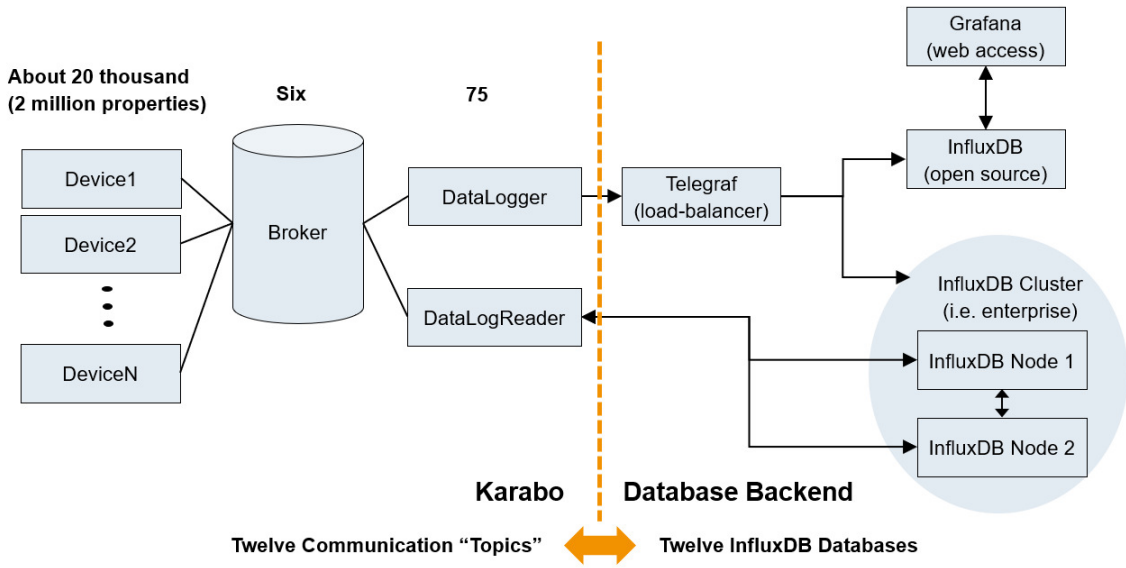


Figure 4: Schematic view of Karabo data logging and its InfluxDB backend.

installations, a new logger and a new log reader device were developed. The logger manager device is now configurable to switch between ascii file and InfluxDB backend.

An InfluxDB installation can host many independent *databases*. All data of a Karabo installation is stored in one such database, identified by the broker topic of the Karabo installation. Each device is mapped to three InfluxDB *measurements*, one for the schema, one for events like device instantiation and shutdown, and the main one for the device properties. InfluxDB *fields* relate to Karabo properties. To ease schema evolution of devices like changing the data type of a property, the field name is suffixed with its type, e.g. `actualPosition-FLOAT`. The mapping between the Karabo and InfluxDB supported types is straight forward for integer or floating point numbers, booleans and strings. Special treatments are needed for the following cases: Non-numeric floating point values (`nan`, `inf`) are stored as strings in fields with an extended field name suffix (e.g. `-FLOAT_INF`), unsigned 64-bit integers are reinterpreted as signed integers, and newline characters in strings have to be mangled. Since InfluxDB does not support arrays, vectors of numbers are stored as comma separated strings. To overcome the ambiguity between an empty vector of strings and a vector with a single empty string, vectors of strings are stored as the base64 encoded JSON representation. Only the Karabo specific data types like raw binary data, table data, and the schema require Karabo for de- and encoding. These types are stored as base64 encoded result of Karabo's binary serialisation.

Many devices update their schema regularly at run time, but typically only a few schema variants exist. Since the serialised binary of a schema (i.e. the self-description of a device) can easily surpass 500 kB, only a schema digest is stored for every update. The schema itself is only stored when the value of such a digest is not yet stored in InfluxDB.

Property updates are transmitted from the data logger device to database backend using the InfluxDB line protocol. By default, data is flushed when at least 200 lines are accumulated.

Figure 4 shows a schematic overview of the Karabo data logging infrastructure with its InfluxDB backend. About 20,000 devices with roughly 2 million properties are distributed among twelve Karabo installations that communicate via six broker instances. In total, 75 data logger devices write to a single load-balancer. The balancer duplicates and forwards the data, once to a standalone InfluxDB instance running the open source edition, and once to a cluster of two instances for data and service redundancy. Running instances in a cluster requires the InfluxDB enterprise edition. In case one of the two storage locations is down, the balancer caches the data in memory. The cluster is the main backend, serving Karabo initiated read requests, whereas the open source instance can be used by other services, without interfering with the control system.

Performance and Operational Experience

InfluxDB data logging at the EuXFEL is in production since summer 2020; data from January 2020 onward has been migrated into the new system. So far, more than 240 billion property updates have been stored, increasing by about 10 billion each month. In total, about 14 TB of disc space is needed per InfluxDB node so far.

The network input to the load balancer is typically about 20 Mb/s, but varies, reflecting that Karabo is event-driven and that the list of Karabo devices that are online at any given point in time can vary. Usually, the data is available for reading with a delay of about 30 s only. This performance was significantly compromised when a Karabo device injected data with timestamps months in the future. Due to the internal data layout in InfluxDB *shards* containing temporal blocks of data, the database frequently internally reorganised

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

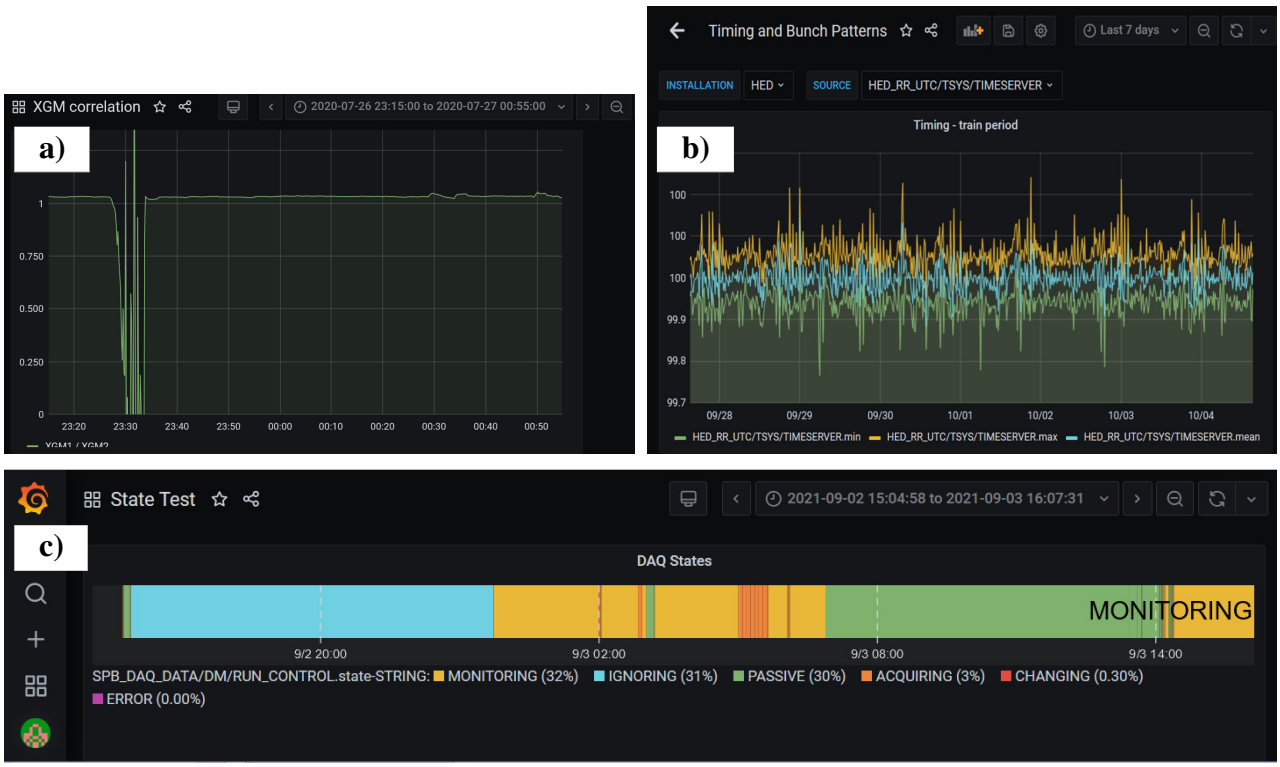


Figure 5: Examples of data displays using Grafana: The ratio of properties from two different devices (a), an overlay of minimum, mean, and maximum of a data trendline that has too many points to show them all (b), and the visualisation of the time development of a string property (c).

data, and the latency increased to over an hour. If there are too many points in a requested time span, down-sampling of trendline data is done in InfluxDB by querying averaged data in equally spaced time intervals. Therefore, short-term features are better preserved and data requested for only slightly different time intervals is more uniform.

All the data stored so far (> 1.5 years) is quickly available. A GUI request to display one week of data of a property that updates every 5 s (and therefore needs averaging) is almost instantaneous.

The planned 3 years of data retention will be affordable. In addition, the InfluxDB internal organisation of the data in shards offers the possibility to keep a reduced set of the data (e.g. averages) for even longer.

GRAFANA USER INTERFACE

Using InfluxDB instead of a custom data storage backend allows external tools to query, retrieve and visualise the data. At the EuXFEL, Grafana [8] has been chosen due to its rich feature set, the fact that both Influx query languages, InfluxQM and Flux, are exposed, and due to a large and active online community that provides many examples.

The resulting low entry threshold enables non-developers to create data views in so-called dashboards, available for others. A nice example of a simple data display that is not easily available within Karabo is the correlation of properties of different devices, see Fig. 5a). When down-sampling is needed, further statistical options besides averaging are available, e.g. minimum and maximum value within the evaluated interval. Overlaying all three as in Fig. 5b) allows allows efficient visualisation of trends alongside outlier preservation. Grafana can even be extended by plugins. The one used in Fig. 5c) displays the time evolution of a string that changes between a few discrete values.

Besides usage in the instrument hutches, Grafana dashboards have become a key element of the Data Operation Center (DOC) at the EuXFEL. The DOC is a co-effort of EuXFEL's Data Department groups: Controls, Electronic and Electrical Engineering, Data Analysis, IT & Data Management, and Detector Operation. The DOC monitors services that the department provides during X-ray operation, and gives pro-active support for the scientific instruments. The main Grafana dashboard of the DOC gives an overview of the status of the most important services and displays alerts if something is outside its expected range, e.g. the frame rate of centrally triggered cameras, see Fig. 6. In this example, the status overviews show no errors whereas the DOC alerts show some long lasting, non-critical problems[1]. Scientific data taking and detector calibration processing are followed closely. For similar systems placed at several instruments, the pull-down menu of a generic dashboard can be used to switch between the different installations, enabling the

[1] Further fine tuning is needed to avoid their appearance on the main page.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

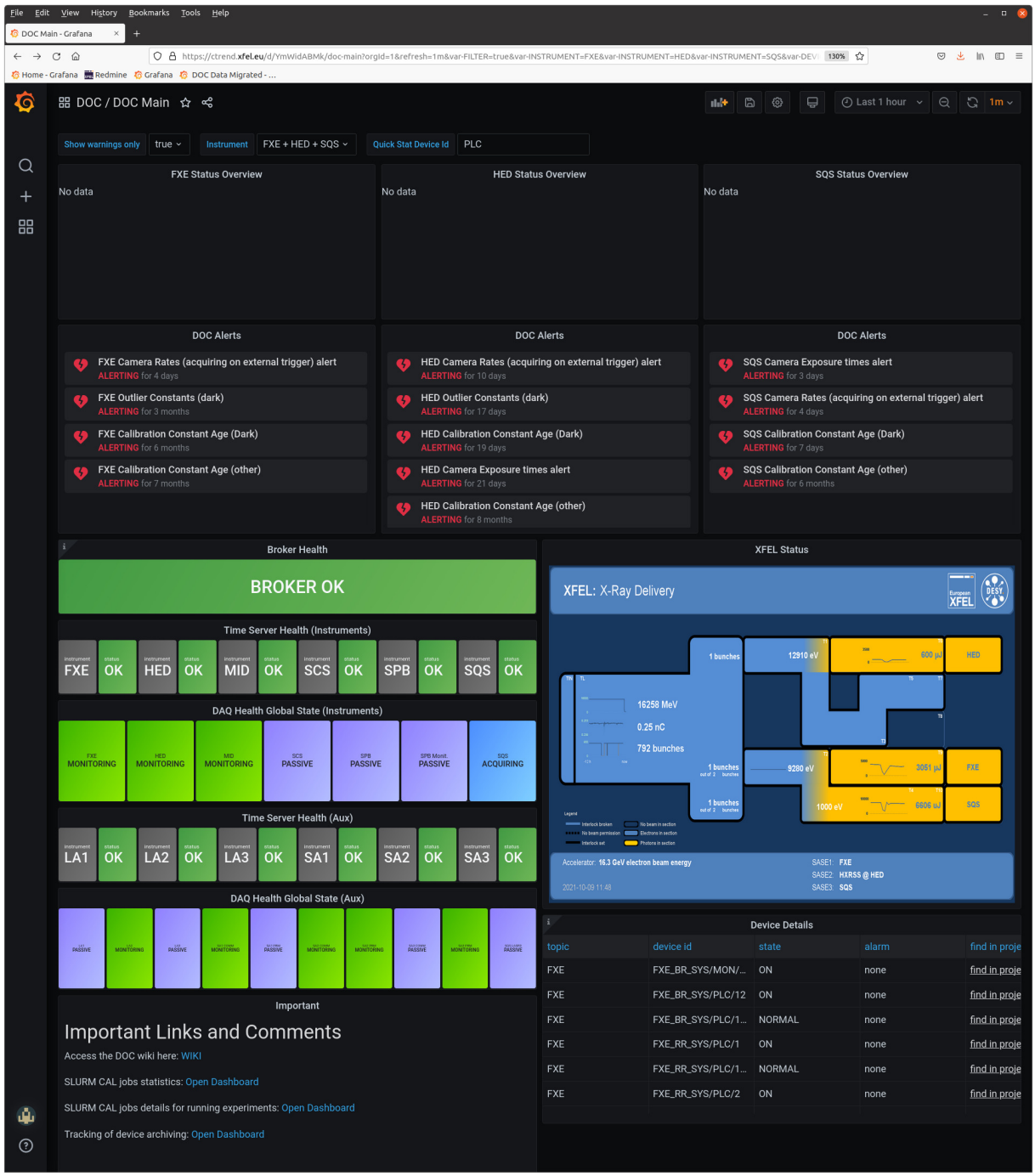


Figure 6: The main Grafana dashboard of the EuXFEL Data Operation Center (DOC), selected to focus on the HED, FXE and SQS instruments.

DOC shift crew to change focus, depending on the running scientific programme, but keeping a unified layout.

CONCLUSIONS

The event-driven, broker-based control system Karabo is used to operate the photon tunnels and scientific instruments at the EuXFEL. Its first ascii file based data logging system was able to fulfil the initial needs, but the longer the facility operated, the clearer it became that the file based backend does not scale well with long term operation. Furthermore, the poor down-sampling capability sometimes confused operators interested in the past state of their system.

A new storage backend using InfluxDB has been developed and is now in operation at the EuXFEL since summer 2020. The goals to overcome slow responses and the lack of statistical methods for data read-back have been met.

Even more, the storage in a community-driven backend allows using a well developed data display and analysis tool,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL04

Grafana, without interference with the control system. Its wealth of features and online examples made Grafana dashboards a key ingredient of the new EuXFEL Data Operation Center that focuses the support of the scientific programme given by the EuXFEL Data Department.

All together, the efforts to interface Karabo data logging with InfluxDB and to setup a reliable database infrastructure have surpassed the expectations to improve the user experience when interacting with historic data in the Karabo control system.

REFERENCES

[1] M. Altarelli, "The European X-ray free-electron laser facility in Hamburg," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 269, no. 24, pp. 2845–2849, 2011.

[2] B. C. Heisen *et al.*, "Karabo: An integrated software framework combining control, data management, and scientific computing tasks," in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, (San Francisco, CA, USA), JACoW Publishing, Oct. 2013, pp. 1465–1468.

[3] S. Hauf, B. Heisen, *et al.*, "The Karabo distributed control system," *J. Synchrotron Rad.*, vol. 26, pp. 1448–1461, 2019, ISSN: 1600-5775. DOI: 10.1107/S1600577519006696.

[4] G. Flucke *et al.*, "Status of the Karabo control and data processing framework," in *presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, (New York, NY, USA), JACoW Publishing, Oct. 2019, p. 938.

[5] InfluxData Inc., *InfluxDB*, https://docs.influxdata.com/influxdb.

[6] *Prometheus*, https://prometheus.io.

[7] Timescale, *Timescale*, https://www.timescale.com.

[8] Grafana Labs, *Grafana*, https://grafana.com.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

PHOTON SCIENCE CONTROLS: A FLEXIBLE AND DISTRIBUTED LabVIEW™ FRAMEWORK FOR LASER SYSTEMS

B. A. Davis*, B. T. Fishler, R. J. McDonald
Lawrence Livermore National Laboratory, Livermore, California, USA

Abstract

LabVIEW™ software is often chosen for developing small scale control systems, especially for novice software developers. However, because of its ease of use, many functional LabVIEW™ applications suffer from limits to extensibility and scalability. Developing highly extensible and scalable applications requires significant skill and time investment. To close this gap between new and experienced developers we present an object-oriented application framework that offloads complex architecture tasks from the developer. The framework provides native functionality for data acquisition, logging, and publishing over HTTP and WebSocket with extensibility for adding further capabilities. The system is scalable and supports both single server applications and small to medium sized distributed systems. By leveraging the application framework, developers can produce robust applications that are easily integrated into a unified architecture for simple and distributed systems. This allows for decreased system development time, improved onboarding for new developers, and simple framework extension for new capabilities.

INTRODUCTION

In contrast to large experimental physics programs, small to medium size experiments and test-beds generally have less resources in terms of manpower, funding, and schedule. Developers are often faced with the task of standing up a distributed control system from scratch under tight deadlines with limited personnel. In these situations, it is imperative to choose a programming language that allows for quick hardware integration and prototyping.

Under these circumstances, NI LabVIEW™ software is often chosen for several reasons. First, it has an extensive hardware ecosystem with options for benchtop, distributed, and embedded hardware systems [1]. Interfacing with these systems from the LabVIEW™ development environment is streamlined and there are offerings for systems across the spectrum of determinism. Simple DAQs provide baseline functionality for non-deterministic applications, and soft and hard real-time situations are handled by RTOS and FPGA applications, respectively.

LabVIEW™ software is also attractive due to its shallow learning curve and low barrier to entry for those without a classical programming background. It uses a graphical programming style combined with a "dataflow" paradigm for organizing functionals and variables and defining execution order [2]. Many workflows for data acquisition, analysis, and logging are built in, and examples and documentation abound. It is simple for novice developers or even end users to create a baseline DAQ system to collect experimental data.

The result is an attractive platform for developing small scale experimental systems. In many cases, control system developers need not get involved at all - scientists and operators can quickly develop the skills to work with LabVIEW™ programming. However, this story becomes less clear when moving from small scale systems to medium scale systems with multiple distributed Front-End Processors (FEPs). Leveraging NI hardware remains an attractive prospect, as it eliminates the need for custom RTOS machines or FPGA boards to handle deterministic applications. However, the very advantage of easy software development can quickly become a burden instead.

Simple LabVIEW™ applications are singular in purpose – they interface with a small number of devices, acquire data, perhaps execute a sequence, and log data to disk. This can be accomplished by a novice developer, or even an end user, as previously mentioned. However, more often than not, such simple systems suffer from a lack of scalability and extensibility. Of course, a piece of software designed to control a single experiment has no need for scaling or extension, provided that system requirements are well-defined before the development begins (a tall assumption, but one we take for granted here).

The disconnect arises when taking similar software development practices and applying them to a larger scale, distributed system. While developing LabVIEW code to control a single system can be accomplished by those with little to no previous software engineering background, developing a distributed, extensible, and scalable system for a larger system requires more experience, skill, and rigor.

Often for a simple system, there is a single developer who creates an application to run the experiment. But for systems of increased complexity, multiple developers of varying skill levels must work together to create a series of interconnected applications across a number of FEPs. In such a situation, a unified architecture must be developed to ensure scalability across the system. Similarly, extensibility becomes key to adding new capabilities over time as the system evolves, as a larger scale system will likely be in operation for longer than a small testbed.

Thus the ideal architecture for developing LabVIEW™ applications for mid-scale distributed control systems must be scalable for any number of devices and FEPs, extensible for adding capabilities across the lifetime of a project (and ideally to future projects as well), and – most importantly – accessible to developers at all skill levels.

To accomplish these goals, we have developed an object-oriented distributed architecture for LabVIEW™ applica-

* davis287@llnl.gov

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

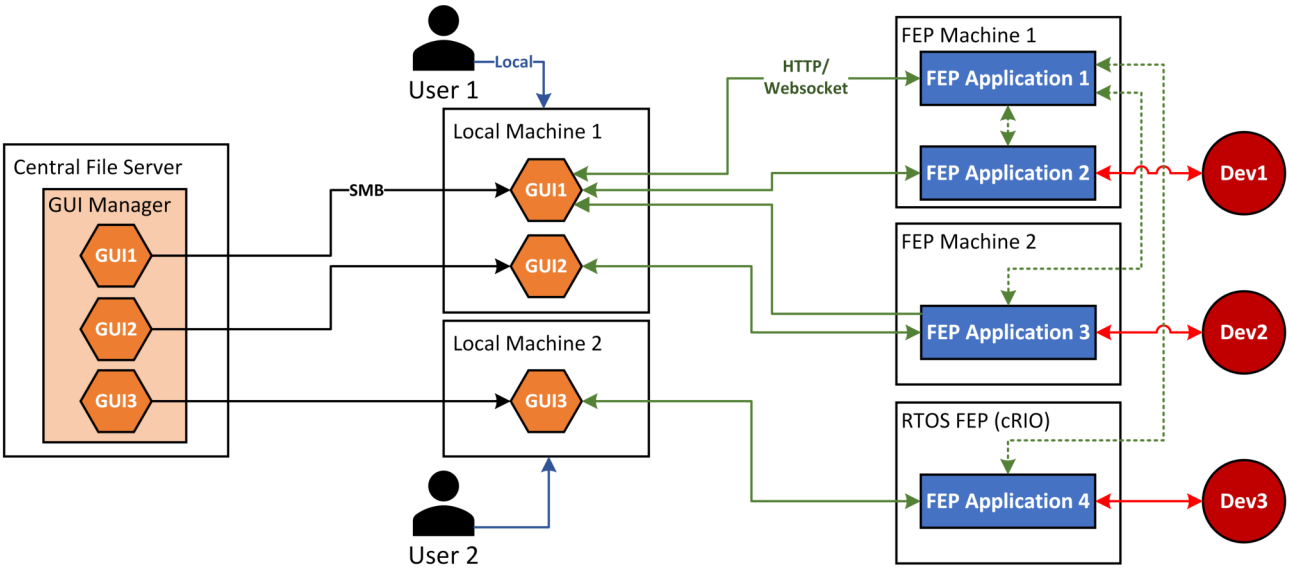


Figure 1: An example system architecture for a small project. Arrows represent communication over the LAN. Black arrows represent Server Message Block (SMB) protocol (mapped network drives), blue arrows represent local (mouse/keyboard) interaction, green arrows represent HTTP or Websocket connections, and red arrows represent device communication.

tions entitled Photon Science Controls (PSC). The architecture leverages inheritance and polymorphism to provide an extensible framework with discrete, encapsulated components. It natively provides flexible inter-application communication over HTTP and Websocket to support systems distributed over a Local Area Network (LAN), with capability to support other protocols as well.

Perhaps most importantly, it is designed to offload common software tasks from developers so that they can focus on discrete functional blocks. Tasks such as intra-application communication, logging, system health, and the aforementioned inter-application communication are pre-packaged and separately source-controlled. This ensures that the architecture remains accessible for novice and experienced users alike, while guaranteeing that functionality remains equivalent across all system applications.

SYSTEM ARCHITECTURE OVERVIEW

Systems using the PSC architecture are designed as peer-to-peer systems composed of FEP applications and client applications. FEP applications are designed to handle device interfacing, sequencing, supervisory logic, and real-time calculation. These are the applications that use the PSC architecture. Client applications include GUIs, real-time status verifiers, and data displays. FEP application peers can communicate with any other FEP application on the network. Client applications can also communicate with any FEP application, but do not connect with each other. Figure 2 shows a example representation of this network architecture.

Combining FEP applications and client applications allows for the development of a distributed control system with user access available from any connected terminal. Figure 1 shows an example distributed system designed under this paradigm. The example system has four FEP applications distributed across three FEPs. Applications two through four handle device communication, and application one runs some supervisory application that requires data from the other three. The client GUI applications are served up from a central file server, which ensures that they are available from any local machine on the LAN.

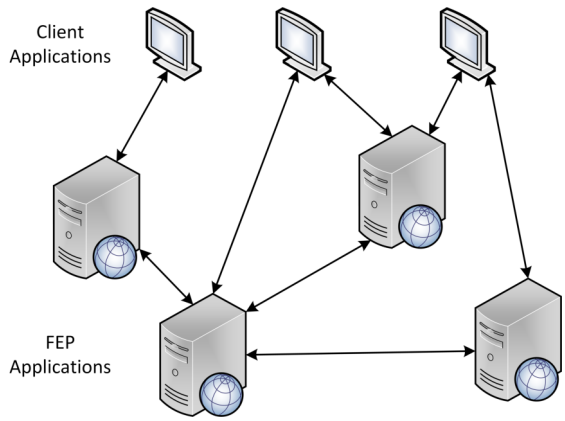


Figure 2: Peer to peer network architecture with FEP applications and client applications.

All communication between client applications and FEP applications, and between FEP applications is done over TCP - specifically using HTTP and Websocket. Users interact with the FEP applications through the remote client GUIs, which can send commands and receive data from the FEP applications. FEP applications can manage multiple connections from clients and peers, allowing for simultaneous monitoring from multiple local workstations. This is espe-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

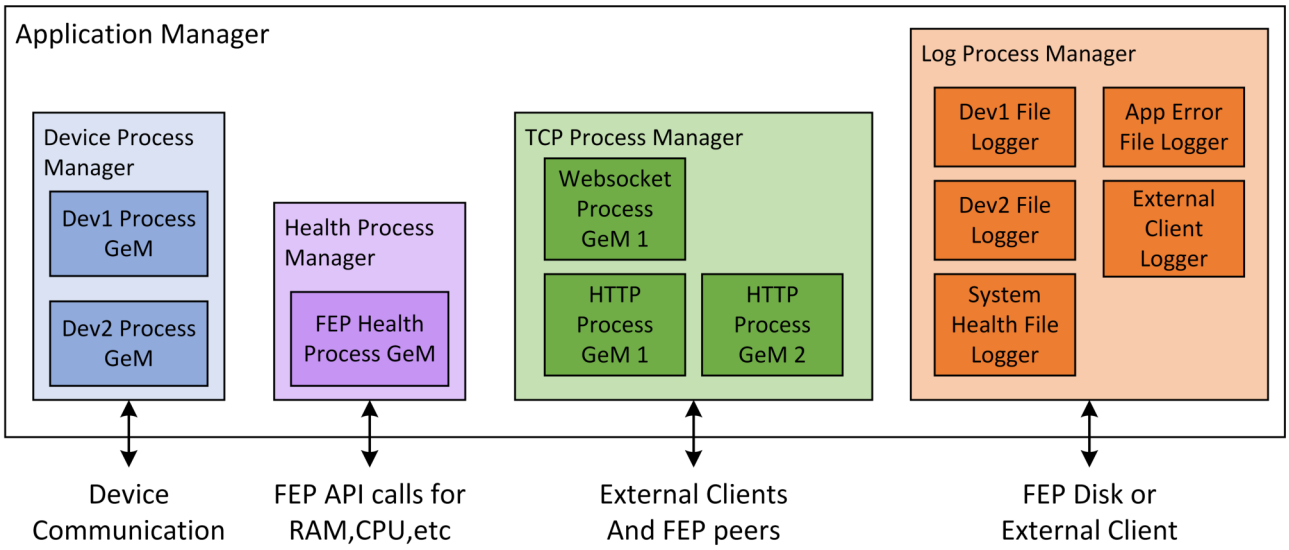


Figure 3: Sample FEP application with hierarchical architecture. Each Manager and Process is a self-encapsulated process running asynchronously.

cially important to support system designers and operators through the commissioning, qualification, and operational lifetime of the system as UI and workstation needs change.

The flexibility of the PSC architecture allows for it to be used on any target with a LabVIEW runtime. This includes Windows machines for non-deterministic applications, and NI hardware targets such as cDAQ and cRIO embedded processors for RT applications. These hardware devices run Linux RT as their RTOS, and cRIO targets also include an FPGA layer for hard real time applications. This allows the architecture to cover the full range of determinism needs for the system without requiring rework or adaptation.

FEP APPLICATION ARCHITECTURE

Application Architecture Overview

FEP applications form the core of the distributed control system and cover most of the functional requirements of the project. They communicate with devices; acquire, process and log data; broadcast and request data from other peers; and handle communication with client applications to send and receive data and commands from users.

The functions of a given FEP application are broken into discrete chunks called Generic Messengers (GeMs). Each GeM is a self-contained asynchronous process that handles a defined subset of the FEP application functionality. GeMs in an FEP application are launched in a manager-worker hierarchy. There is a top-level Application Manager GeM, one to many mid-level Process Manager GeMs, and one to many Process GeMs. Process GeMs control the core behavior and provide the main functionality of the application. Process Managers and the Application Manager allow for data roll-up from the Process GeMs as well as high-level application management functionality.

Figure 3 shows an example of this application hierarchy with some typical GeMs. This particular application would communicate with two devices, monitor the health of the FEP (CPU usage, RAM, etc), manage communications with external applications, and log device and application data to disk and an external client like an industrial Historian. Process managers and the Application manager would handle high level application behavior - monitoring the health of different processes, managing processes closing or being instantiated, etc.

There is no defined limit to the number or type of GeMs present in a single application. Different classes, devices, protocols, and loggers can be supported to accomplish the purposes of the application as long as they conform to the structure of the GeM ancestor class.

Flexible Functionality Distribution

Intra-application communication between GeMs is a microcosm of the peer to peer inter-application communication scheme of the distributed system. Each GeM is individually addressable so that data can be passed around the application as necessary. For example, the Dev1 CSV File Logger in Fig. 3 would subscribe to data from the Dev1 Process stream to log it to disk. Like the peer-to-peer relationship between FEP applications at the system level, any GeM can get data from or issue commands to any other GeM in the FEP application.

This nested approach to communication allows for maximum flexibility in defining application and system organization. Depending on system requirements, FEP applications can be organized by functionality, device type, or logical subsystem grouping. GeM processes that fulfill the system requirements are then distributed across the FEP applications to match these functional groups. System hardware archi-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

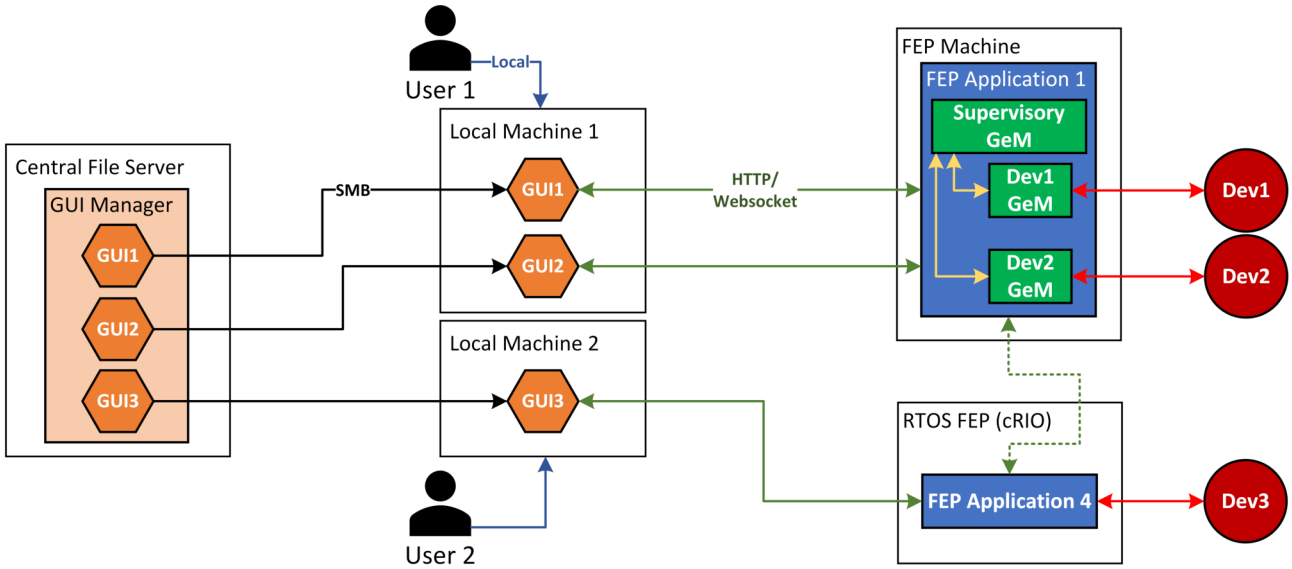


Figure 4: Redistributed architecture from Fig. 1. Note that the functionality of applications two and three has been moved into GeMs in application one. Yellow arrows denote internal communication (queues). This architecture is functionally identical but may provide different performance characteristics that would prefer its usage. Not pictured are the GeMs handling the external communication.

tecture, performance requirements, and logical delineations can all drive these decisions.

For example, Fig. 4 shows how the example system in Fig. 1 could be rearranged. We have consumed the functionality of FEP applications two and three into FEP application one, and instead used the internal data lines for the supervisor rather than external (HTTP, Websocket) to get their data. Note that FEP application four remains on a different FEP, so FEP application one still uses external communication for its data.

This reorganized system would perform the same tasks as the architecture presented in Fig. 1, but may provide performance benefits that would make it preferable. In short, a PSC2 system architecture provides the capability to tune the level of distribution to match physical and performance needs at the macro "distributed system" level and the micro "application GeM" level.

Class Hierarchy and GeM Functionality

Figure 5 shows a class diagram for the Generic Messenger ancestor class. Note that most GeM classes are descendants of the Process class, as that is where the core functionality of an application is executed. However, Process Managers and the Application Manager also inherit from the GeM ancestor class, which means they inherit similar functionality to Process classes.

The functionality provided by the GeM ancestor includes five key items. The ancestor provides a common process main that ensures these five items are all implemented identically across all descendant GeMs.

Data polling / creation Every GeM includes a method to poll and create data at a predefined rate. For device process classes, this would include the device status data. For other GeMs, this provides an internal status. As an example, a logger class would provide information on its logging speed, how many items are left in the logging buffer, the current file, etc. Data is generically typed in a key-value structure for easy handling across GeMs and peer applications.

Command Handling Every GeM has a command handler that allows it to receive messages from other GeMs in the system. This is the method by which data is passed throughout the application. A command message is usually bundled with a response queue, which allows for a response to be sent back upon command action completion.

Lossy and Non-Lossy Data Streams After the GeM data is produced in the polling loop, it is piped into two data streams. The first, a lossy stream, is a single element queue. The element is overwritten upon each polling cycle, hence the designation as lossy. This is what the command handler can use to report the latest data sample upon request. It is often used for discrete requests, like by a TCP process for sending data to a GUI.

The second data stream is a non-lossy stream. Data is pushed into a buffered queue so that each sample is captured. This is the preffered method for passing data to loggers or other processes that need a complete data history. GeMs can subscribe to this data stream by sending a subscription request over the command handler.

Thresholding and Alarming Particularly for device process GeMs, it is important to monitor data values for

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

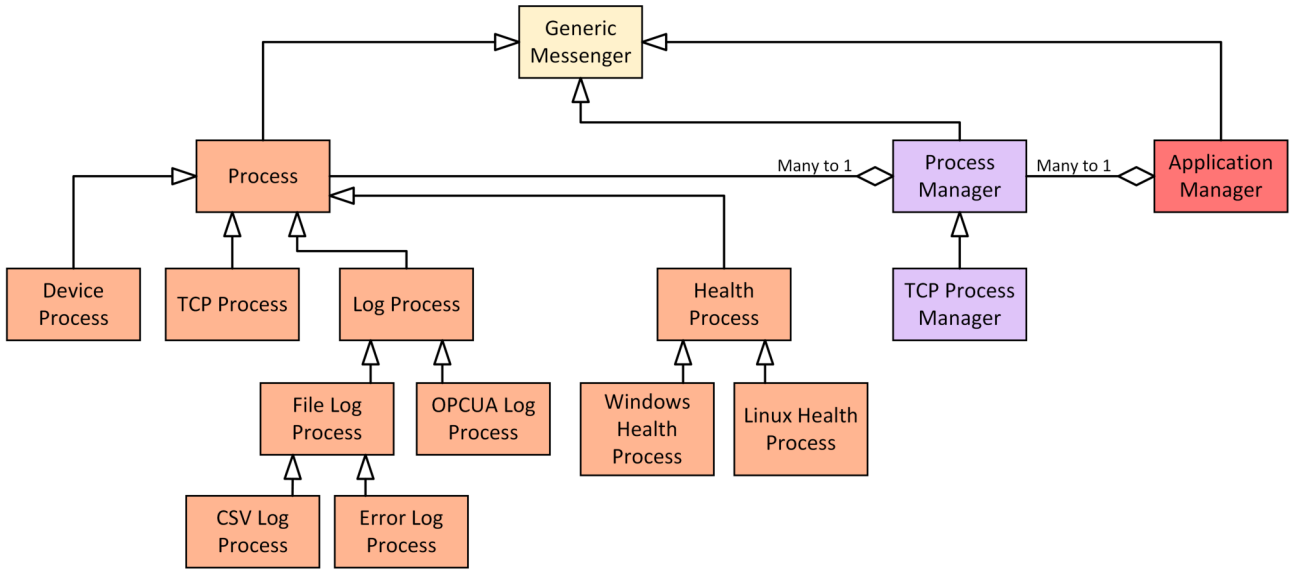


Figure 5: Generic Messenger Class Hierarchy. Most available architecture GeMs are of the "Process" descendancy, as this is where functionality is most heavily defined.

anomalies and take actions if they exceed pre-defined levels. Every GeM has a built in threshold handler that allows for definition of levels and actions to take based on named data values. These thresholds can be numeric or boolean. Based on the level of the data under review, UI warnings and alerts or Machine Safety System (MSS) actions can be taken.

Internal Error Handling Occasionally GeMs will encounter an internal error and self-terminate. The error handling built into all GeMs ensures that these errors are logged to disk for review. This is essential for addressing issues that occur during commissioning and operations.

GeM Lifeline and Application Startup

Each GeM in the system goes through a similar lifeline. They are instantiated and initialized, then spun off to run an asynchronous process main. They are then terminated upon task completion or application end and run a cleanup routine.

PSC uses a Factory style object creation scheme [3] upon application startup. When a PSC application is loaded, an application factory is created. This factory object (not a GeM) loads a configuration file that defines the objects needed for the application to run. It then loads the necessary classes into memory and instantiates them.

The launch process at application startup is recursive in nature, based on the application hierarchy. The factory creates an application manager GeM, within which are a number of process manager GeMs who each contain a number of process GeMs. GeMs are then launched from the bottom of the hierarchy to the top – processes are each spun off, then the corresponding process managers, and then finally the application manager. This ensures a consistent boot order that is defined in the application configuration. A defined boot order is important to ensure that GeMs that depend on other GeMs are initiated in dependency order.

Not all GeMs are initiated at application startup. It is entirely possible for GeMs to be initiated by events that occur during application operation. As an example, a TCP Process Manager (see Fig. 3) listens for external connections and instantiates TCP processes to handle them during runtime. These processes persist until the connection is closed or timed out, and then self-terminate.

Data Recursion

All GeM data is wrapped up recursively through the application hierarchy (Process -> Process Manager -> Application Manager) (see Fig. 6) In other words, each Process Manager's data packet contains all the data packets of its worker Processes, and the Application Manager's data packet contains all the data packets of its worker Process Managers. This allows for an external client or peer to get all the data from an application by making a single query to the top-level Application Manager.

Addressing and External Communication

As previously mentioned, GeMs can communicate with other GeMs in a specific FEP application, and FEP applications can communicate with other peers and client applications. Every FEP application in a distributed PSC2 system has an IP address and associated TCP ports. Furthermore, each GeM within an application is assigned a URL based on the application hierarchy. This URL serves as an address for both external and internal communication. URLs are generated by the following pattern: **IPaddress:port/ApplicationName/ProcessManager/Process**.

As an illustrative example, **185.20.20.1:5437/SampleApplication/DeviceManager/Device1** would refer to the Device1 process of the Device Manager Process Manager on

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

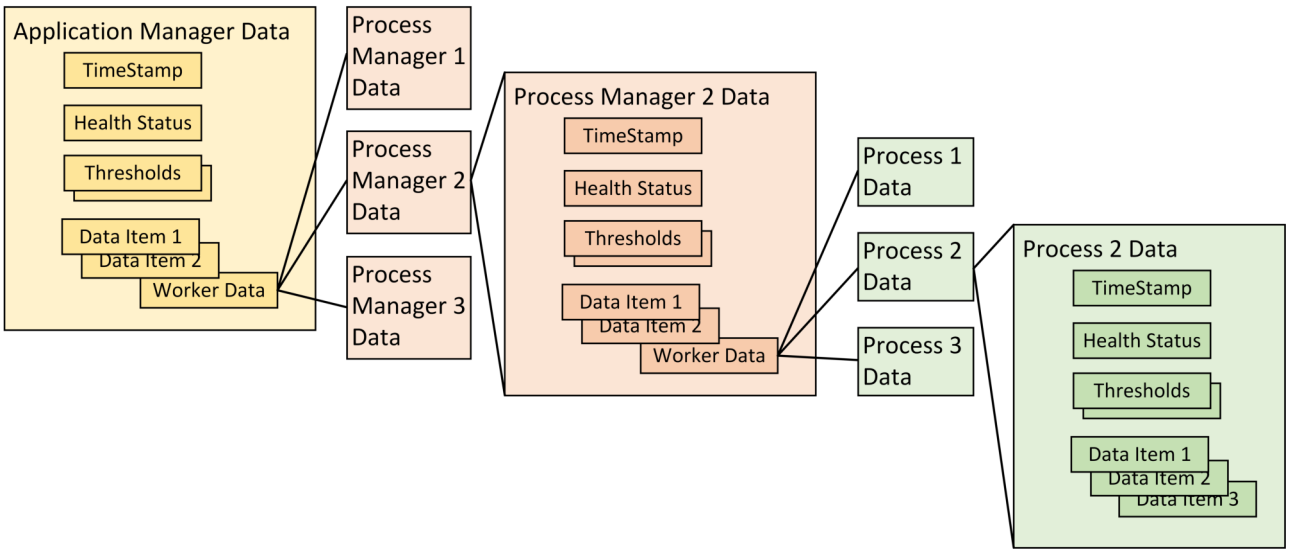


Figure 6: Data Recursion in FEP applications. The top-level application manager contains all the data from the GeMs in the application.

the Sample Application hosted on 185.20.20.1. The port 5437 may be the port setup for HTTP, Websocket, or some other protocol.

The same URLs are used as identifiers for the command queues of each GeM in the application. GeM peers send commands to others' command queues by name. Thus, the unique identifier URL for a GeM serves as a routing path for both external and internal communication.

External communication is routed through a TCP Process Manager GeM, which instantiates worker TCP Processes, one for each connection. Figure 7 shows the dataflow for connecting to an external application. The TCP Process Manager maintains a listener on the specified TCP port. Upon receiving a connection request, it instantiates and launches a TCP Process GeM. This GeM establishes the TCP socket connection with the external application. Messages are sent from the external application to the TCP Process, which then forwards them to other GeMs by URL. The TCP Process GeM persists until the connection closes or times out, at which point it self-terminates.

The TCP Process Manager can instantiate any number of TCP Process GeMs to support multiple connections. These TCP Process GeMs can be HTTP connections, Websocket connections, or other TCP protocols. There is a single connection for each external application, which ensures that data is not corrupted by multiple writers/readers on a single socket.

Application Configuration and User Sets

Configuration items in a PSC application exist across three levels of accessibility - "hard-coded" (lowest accessibility), "static," and "configurable" (highest accessibility). Hard-coded items are configuration items that cannot be modified without rebuilding the application. These are protected by source control and are collected in an Application Configuration file. Static items can be modified without rebuilding the application, but are only applied on startup. Configurable items can be loaded and saved during runtime. Both static and configurable items are collected in User Sets.

Application Configuration File Every GeM maintains a list of configuration items that define its behavior. These form the basis of the Application Configuration that defines the GeMs included in a specific application build. The Application Configuration is loaded upon application startup, and is used by the Application Factory to instantiate all the GeM objects of the application. It lists the participating GeMs, and their configuration parameters that define their behavior. These can include polling rates, timeouts, links to other GeMs, etc. These values are considered hard-coded because they define the behavior of the final application. Because of this, application configuration files are protected as source code, and require rebuild upon changing.

User Sets Device Process GeMs maintain a series of separate configuration files known as User Sets. User Set files allow operators to save and recall device states in order to facilitate different modes of system operation. As with application configuration items, each Device Process GeM maintains a list of user set items. Each item in a user set list is tagged as "startup only" or not. This tag is what separates static and configurable items. Static items are only applied when the application is restarted, so changes to such items in a user set will not be applied immediately. Configurable items, on the other hand, can be recalled at any time during operations.

As an example, imagine a flow controller that is handled by a Device Process GeM. Perhaps there are some warning/alarm thresholds associated with the device that we would like to have the flexibility to change without rebuilding the application (i.e., they cannot be hard-coded in the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

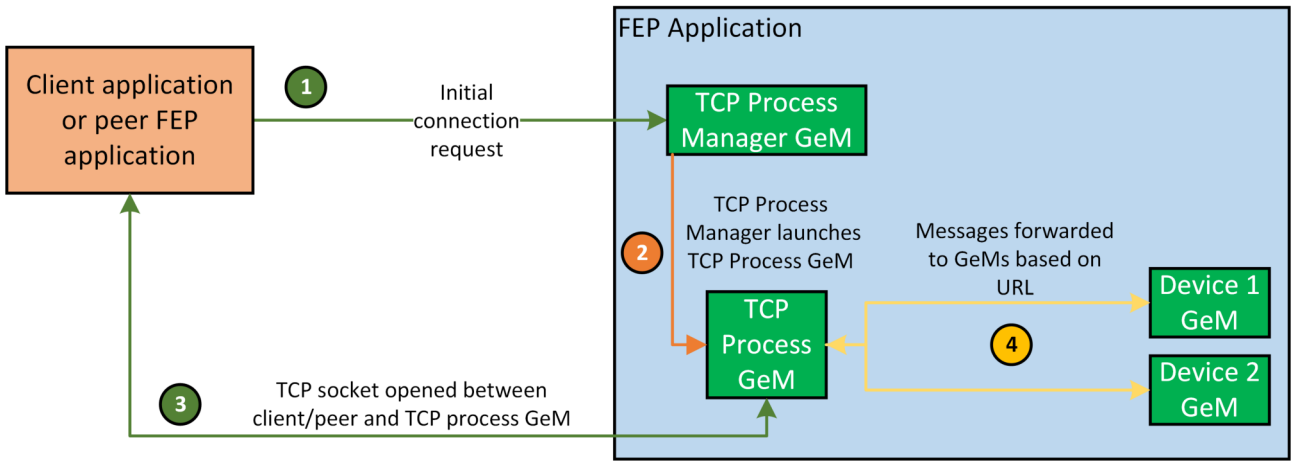


Figure 7: External Communication Process. The TCP Process Manager listens on the specific port for an initial message. Once the message is sent, the Process Manager instantiates a TCP Process to handle the socket connection. The TCP Process persists until the connection is closed or times out. Commands are sent by the client or peer to the TCP Process GeM, which forwards them to their destination based on URL.

application configuration). However, we do not want them to change without review and management approval. These would be an appropriate candidate for static (startup-only) behavior. We may also have a setpoint for the flow controller that needs to change based on what phase of an experiment is currently running. This may be a good configurable item - we can save one user set for each phase, and recall the appropriate one at runtime.

User sets are organized by name, and can be saved and loaded from client UI applications. This gives operators flexibility in defining, modifying, and managing the user sets that need to be applied for continuous operation. Table 1 summarizes the three types of configuration items.

Table 1: Configuration Item Summary

Type	Associated File	Accessibility
Hard-Coded	Application Configuration	Rebuild Application
Static	User Set (startup-only tag)	Restart Application
Configurable	User Set	On Demand

CONCLUSION

The use of the PSC architecture helps to bridge the gap between small and large scale experimental systems by providing a scalable and extensible framework for rapid development timelines. Encapsulating functionality into interconnected Generic Messenger objects allows for unlimited flexibility in defining the distribution of system requirements over the system LAN. By leveraging inheritance and polymorphism, base functionality is preserved across all descendant GeM classes, providing a simple workflow for developing new features and capabilities.

By abstracting functionality into the classes that form the architecture, developers can focus solely on developing code that fulfills system requirements without having to deal with high level processes like communication and logging. Taken all together, the architecture provides an excellent solution for medium scale distributed systems and projects with funding, time, and personnel limitations.

ACKNOWLEDGMENTS

The authors would like to acknowledge the work of Daniel Smith, Chen Lim, and Sridhar Kuppuswamy for their work on the previous iteration of the architecture.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Released as LLNL-CONF-827711.

REFERENCES

[1] Designing Control Applications with Data Acquisition Hardware and NI-DAQmx, https://www.ni.com/en-us/innovations/white-papers/09/designing-control-applications-with-data-acquisition-hardware-an.html

[2] Benefits of Programming Graphically in NI LabVIEW, https://www.ni.com/en-us/innovations/white-papers/13/benefits-of-programming-graphically-in-ni-labview.html

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Creational Patterns: Factory Method," in *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Boston, MA, USA: Addison-Wesley Professional, 1994, ch. 3, sec. 3, pp. 107-117.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

ROMULUSlib: AN AUTONOMOUS, TCP/IP-BASED, MULTI-ARCHITECTURE C NETWORKING LIBRARY FOR DAQ AND CONTROL APPLICATIONS

A. Yadav, H. Boukabache*, N. Gerber†, K. Ceesay-Seitz, D. Perrin
European Organization for Nuclear Research (CERN), Geneva, Switzerland

Abstract

The new generation of Radiation Monitoring electronics developed at CERN, called the CERN RadiatiOn Monitoring Electronics (CROME), is a Zynq-7000 SoC-based Data Acquisition and Control system that replaces the previous generation to offer a higher safety standard, flexible integration and parallel communication with devices installed throughout the CERN complex. A TCP/IP protocol based C networking library, ROMULUSlib, was developed that forms the interface between CROME and the SCADA supervision software through the ROMULUS protocol. ROMULUSlib encapsulates Real-Time and Historical data, parameters and acknowledgement data in TCP/IP frames that offers high reliability and flexibility, full-duplex communication with the CROME devices and supports multi-architecture development by utilization of the POSIX standard. ROMULUSlib is autonomous as it works as a standalone library that can support integration with supervision applications by addition or modification of parameters of the data frame. This paper discusses the ROMULUS protocol, the ROMULUS Data frame and the complete set of commands and parameters implemented in the ROMULUSlib for CROME supervision.

INTRODUCTION

The Occupational Health & Safety and Environmental Protection (HSE) at CERN obliges to the protection of CERN personnel and the public from any unjustified exposure to ionising radiation. The radiation protection group (RP) at HSE has the mandate to monitor the radiological impact of CERN's accelerators and installations by active monitoring and logging of radiation levels at different experimental sites spanning the CERN complex. To facilitate this, the new generation of CERN RadiatiOn Monitoring Electronics, called CROME [1], was developed by the CROME team of the Instrumentation and Logistics (IL) section within the RP group and is responsible for the design, development, installation and maintenance of these specialised radiation monitoring systems. Starting from Long Shutdown 2 (LS2) of the Large Hadron Collider (LHC) in 2019, the older generation of radiation monitors, namely The Area Controller (ARCON) is being replaced by the new CROME devices and will be operational for the Run 3 of the LHC in February 2022. The consolidation of the current generation of RAdiation Monitoring System for the Environment and Safety (RAMSES) monitors by CROME is planned to be completed by the Long Shutdown 3 (LS3) in late 2027.

The CROME devices consist of the autonomous Monitoring Units, Alarm Units, a Junction Box and an Uninterruptible Power Supply. [2–4] The autonomous monitoring units, called the CROME Measuring and Processing Units (CMPUs), consist of an ionization chamber and an electronic readout system. The CMPU can be either a wall-mounted system where the CMPU is directly attached to the ionization chamber or it can be a rack-mounted system where the CMPU is connected to the ionization chamber with a specialized cable. The rack-mounted system is used for monitoring areas with high radiation levels that can damage the readout electronics. In this case, a custom plastic ionization chamber with graphite coating is placed directly into that area with high radiation levels, whereas the readout electronics are placed in an area of lower radiation for their protection. The ionization chamber detects ionizing radiation and converts it to a readable current value. A read-out chip measures this current, which can be within 2 fA to 1 µA. Because these currents can be so low, a specialized cable SPA6 was developed by the RP team at CERN for the rack-mounted system. The SPA6 cable is used for Signal and High Voltage lines up to one kilometer distance. The CMPU's front-end readout interface transmits the current value to the FPGA programmable logic (PL) of a Zync-7000-based System-on-Chip (SoC), which uses it to calculate the real-time radiation dose rate as well as the total radiation dose received in the monitored area at the ionization chamber location. All safety-critical decisions and actions such as measurement, dose rate calculation, temperature compensation, alarm generation and interlock generation are performed by the PL. This is done to ensure system reliability by implementing operations within a Finite-State Machine. A complex programmable logic device (CPLD)-based watchdog works in tandem with the SoC. It monitors the PL state machine to ensure correct dose rate calculations and overall functionality. It is allowed to reset/reboot the SoC when it is in an undefined state. At the detection of dangerous conditions, e.g. if the radiation dose or dose rate exceeds a defined limit, the CMPU automatically generates local and remote alarms and a beam interlock signal that stops the concerned accelerator or machine. Parameters like dose and dose rate limits, current-to-radiation conversion factors, and many more can be configured remotely by the authorized members of the radiation protection group. A schematic of the CROME devices and it's part is shown in Fig. 1.

* Corresponding author Dr. H.B. (hamza.boukabache@cern.ch)
† N.G. is a former CERN Fellow. He is currently at CSEM, Neuchâtel (CH)

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

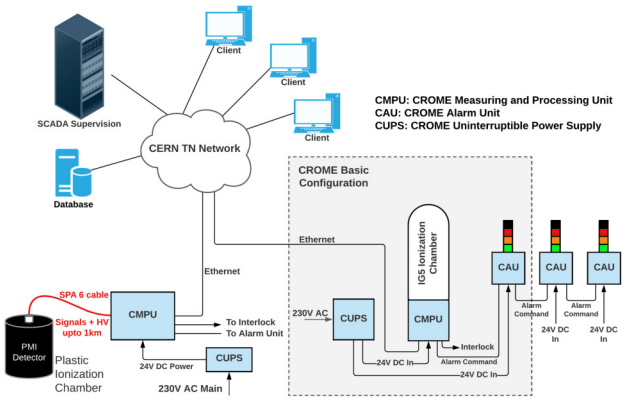


Figure 1: CROME Radiation Monitor components and basic configuration.

The radiation monitoring CROME devices at CERN are required to fulfil safety requirement in accordance with the IEC 60532 standard for radiation protection instrumentation. This is in compliance to the task of interlock triggering of machines in case of high radiation levels. Therefore, CROME devices have been developed to satisfy Safety Integrity Level 2 (SIL 2), as defined in the IEC 61508 standard in accordance with the IEC 60532 for functional safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. We have therefore followed a process and implemented measures to reduce the risk of dangerous failures during the development and operation while following stringent methodology to avoid both random and systematic faults for both firmware [5] and hardware [6].

The CROME hardware deployed throughout the CERN complex communicates with a SCADA supervision system, called Radiation and Environment Monitoring Unified Supervision (REMUS) [7, 8]. In order to facilitate reliable connectivity and logging capabilities, the development of a dedicated TCP/IP-based C networking library, ROMULUSlib was initiated by the CROME team. ROMULUSlib is the communication interface between CROME and REMUS. It sits in the userspace of a POSIX compliant OS and allows multi-user full-duplex communication with CROME devices through REMUS and/or an 'expert application' connected to the devices through the network via Ethernet connection. ROMULUSlib compiles on multiple architectures and at present, ROMULUSlib has been built and tested successfully with gcc 4.8.5 on x84_64, arm32 and with Apple clang version 11.0.3 on x86_64 Darwin MacOS Kernel 19.4.0. In this paper we present the details of ROMULUSlib, the construction of the data packets for command and data communication over the TCP/IP network. We present the overall structure of the ROMULUSlib code base and a set of utilities that are a useful aid for anyone instantiating ROMULUSlib for their DAQ and Control use case. Beside this, an independent project, RomLibEmu [9] was initiated to develop a regression testing framework for debugging systematic faults during development and before the release of every new version of the ROMULUSlib.

This paper is organized as follows: in background and related work, we discuss some of the previous library implementations developed for TCP/IP supervision and how ROMULUSlib compares to the same. In the next section, we give a brief overview of the CROME radiation monitoring system architecture and the REMUS Supervision system, followed by the details of the implementation of ROMULUSlib that forms the interface for the SoC to SCADA communication. In this section we present the details of the TCP/IP frame, supported commands, parameters and ranges with their corresponding acknowledgement response and test utilities. ROMULUSlib supports both real-time stream and buffered data stream such as status request, historical record and events that are discussed here. In the end, we present the current application use cases and tests deployed at CERN and at the European Spallation Source (ESS) followed by the future work and adaption of ROMULUSlib library for varied use cases.

BACKGROUND AND RELATED WORKS

Early control operations at CERN radiation protection employed PLC to SCADA based communication for control application. CERN also uses this for LHC Cryogenics control in accelerator and experimental control, Gas systems, Cooling systems, HVAC etc. For control and monitoring, Ethernet based communication protocols such as Profinet and Ethernet/IP are employed [10]. The UNICOS framework is one example of the same [11]. Another example of the communication protocol used for real-time automation is the MODBUS TCP/IP which uses the MODBUS protocol for communication encapsulated within the TCP/IP wrapper for communication over the ethernet.

CERN currently started very recently using Zynq-based SoCs for a wide range of applications in detector control and front-end data acquisition electronics designs. These SoCs offer greater flexibility to the user for design on programmable logic, offers an embedded linux platform on the processing system for high-level software development and provides faster performance due to the PS-PL interface. The open-source WinCC OA is the currently widely adopted SCADA framework for detector control applications which is also used by REMUS supervision [8]. One of the adopted communication frameworks in ATLAS Detector for DAQ to DCS communication is the Quasar framework [12] which allows building of OPC UA servers for client-server communication. It is employed at ATLAS's Tile Calorimeter measurement[13] and Global Feature Extrator (gFEX) Hardware Trigger[14] among others. Quasar's client-server communication model integrates with the WinCC OA SCADA and allows read/write of predefined variables, and creation of multiple OPC servers. A new library, MilkyWay, based on FreeOpcUa is further being developed in Python [15].

While PLC-to-SCADA based communication frameworks exist; with the introduction of SoC-to-SCADA systems, fast and reliable safety critical communications framework needed to be developed. Therefore, the development of com-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

munications protocol for safety-critical radiation monitoring and control was initiated in 2015 by the CROME team at CERN. The communications specifications provided by the ROMULUSlib allows user to have communication reliability of TCP/IP while having greater control over creation of variable length data frames for DAQ and control applications. ROMULUSlib runs on each communicating device and communicates with the WinCC OA based SCADA through a driver. The users have the flexibility to update their own parameters of interest which compiles seamlessly with ROMULUSlib. This further simplifies integration with embedded linux userspace application development which makes ROMULUSlib suitable for adoption for SoC-to-SCADA TCP/IP communication systems.

CROME: RADIATION MONITORING ELECTRONICS AT CERN

The CROME device is developed to provide a versatile interface for:

1. Continuous real-time monitoring of ambient dose equivalent rates over up to nine orders of magnitude.
2. Alarm and interlock functionality with a probability of failure down to $10e^{-7}$.
3. Long term permanent and reliable data logging by linking to a SCADA supervision or an expert application running on a portable PC via Ethernet.
4. Edge computing with powerful processing capabilities for embedded calculation.

The basic configuration of the CROME device consists of the three main parts: CROME Measuring and Processing Unit (CMPU), the CROME Alarm Unit (CAU) and the CROME Uninterruptible Power Supply (CUPS). These are connected to the global supervision structure as shown in Fig. 1. The CMPU hosts the System-on-Module board which implements a Zynq-7020. This SoC is composed of the 32-bit ARM Processing System (PS) which runs CROMiX-18 a custom embedded Linux distribution developed using the YOCTO project and the Programmable Logic (PL) FPGA fabric that is programmed with a parameterizable hardware implementation for readout, calculations and safety-critical controls.

The front-end electronics is also part of the CMPU. It performs analogue-to-digital conversion of the current signal generated by the radiation detector to which it is attached providing a continuous real-time measurement of ambient dose equivalent rates. The CMPU in turn generates radiation alarm and interlock signals and enables long-term and data logging by integration with REMUS. The communication interface for the CROME devices and the REMUS supervision is the ROMULUSlib and it provides four main functionalities:

1. Networking: All communications happening over TCP/IP frame is handled by the functions defined in ROMULUSlib.
2. TCP/IP frame construction: ROMULUSlib provides structs to encapsulate different data variables with support for nearly all C data types. This can be easily updated by updating the struct table. The construction of ROMULUS frame is handled by the ROMULUSlib, including functions to access and modify specific parts of the frame which provides much more flexibility and control to the user.
3. Multiple Communication Modes: ROMULUSlib provides full-duplex communication over TCP/IP via multiple communication modes to transmit single, multiple and infinite frames which can be used for streaming data in multiple ways as per application requirement.
4. Utilities: Multiple utilities are provided within the ROMULUSlib for utilities such as log reporting of application flow, warnings and errors, functions to print frame, struct and internal data struct, checksum check functions and check functions for memory allocation and free memory.

CERN's REMUS supervision [7, 8], is a WinCC OA based SCADA supervision system which is used for communication and parameterization of CROME devices. REMUS integrates the SCADA system with the open-source streaming platform Apache Kafka, a widely adopted technology, that allows data-streaming in near real-time data to the Data Visualization Tools and Web Interfaces. REMUS provides a secure interface for full-duplex communication interface with external Control devices. A functional architecture of the REMUS supervisory control along with the application of ROMULUSlib for data streaming and control from WinCC OA is shown in Fig. 2.

ROMULUSLIB

The CROME devices communicate with REMUS using the full-fledged TCP/IP protocol which is implemented in ROMULUSlib, a stand-alone TCP/IP networking library developed in C that supports cross-compilation and portability by making use of the POSIX-standard, more specifically sockets. TCP/IP communication has been chosen to ensure high reliability and automatic data reordering. Therefore, all data communicated as ROMULUS data packets are encapsulated in TCP/IP frames. The device responds to most of the messages sent from the supervision. This is reconfigurable by defining the structure of the ROMULUS frames and defining the response packets as per requirements specifications. For each received message the device returns an acknowledge message. The acknowledgement messages also ensure correct functionality of devices by having unique request-acknowledge pair. Specific error messages are communicated to be returned to ensure that there is a consistent answer from the device (e.g. malformed or unknown message). All packages received by the devices also contain a Sequence ID of the sender. The response or acknowledge message will include this Sequence ID. This ensures that the device can have multiple concurrent users. Besides SCADA

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

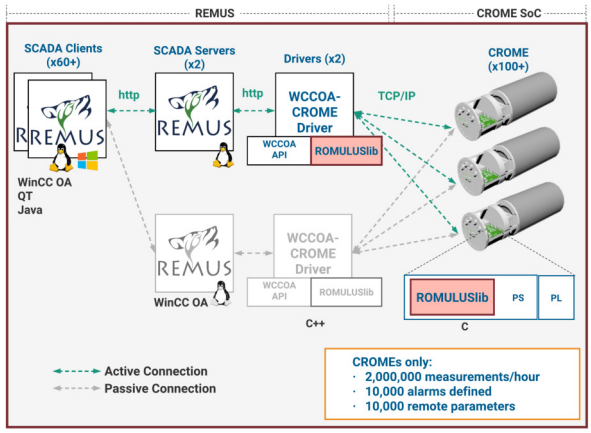


Figure 2: REMUS Supervisory Control and Data Acquisition Architecture [7, 8], Courtesy of Adrien Ledeul.

supervision, the device is also fully configurable and parameterizable by a dedicated standalone 'expert application' that runs on a portable computer and is connected either directly to the device or through the network via the Ethernet socket. For this application, the CROME Team has developed a standalone application in NI LabVIEW that can connect to the CROME devices through the network.

In this section, we present the protocol used for the communication between the CROME devices and the REMUS supervision. The protocol is also used to communicate with the NI LabVIEW-based expert application.

ROMULUS Data Frame

The ROMULUS data packet, as shown in Fig. 3, is encapsulated into a TCP/IP frame whose payload is limited to ETHERET_MTU (1500 Bytes) while having no fixed length for data packet itself in order to obtain more flexibility. Within the ROMULUS Data packet, the header is formed by: data length of the DATA, ID of the source of the message, ID of the message, the command code and the number of parameters. This is followed by the Data corresponding to the command code and the footer, which is the checksum.

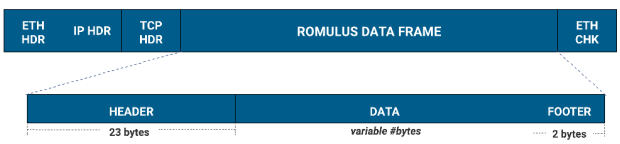


Figure 3: ROMULUS TCP/IP Data Packet.

The maximum data length of the DATA in bytes can be calculated as follows:
*max*DLC = ETHERNET_MTU - [*sizeof*(IP header) + *sizeof*(TCP header) + *sizeof*(romulus_frame_header) + *sizeof*(romulus_frame_footer)]

Both, command and response packets have the same structure. A generic DATA segment of the ROMULUS Data Frame for Get/Set commands is shown in Fig. 4. Note that, multiple parameters can be communicated within one data

Figure 4: Data segment of any generic get/set command in ROMULUSlib.

frame. Therefore, it is possible to initiate multiple operations through a single message. Thus, the variable length of the ROMULUS data frame is due to the field DATA whose length is variable. In the response frame, if no error has occurred, the content of the DATA field of response will begin with an Acknowledgement (ACK) code. Otherwise, if an error occurs, the device returns an error code corresponding to one of the related errors, such as Acknowledgment, Illegal command, Illegal value etc.

We have provided command definition and data frame construction support for Simple and Complex Request-Response Schemes of communication within ROMULUSlib, which are discussed in the following section.

ROMULUSlib Commands

ROMULUSlib commands define how the device and supervision can communicate using the ROMULUS protocol. ROMULUSlib currently supports 18 commands to interact with the CERN's REMUS/ROMULUS protocol. These commands are categorized into two different schemes of communication within ROMULUSlib. These are namely Simple Request-Response Scheme and Complex Request-Response Scheme.

ROMULUS Simple Request-Response Scheme: This is the most common mode for communication using the ROMULUS protocol. The supervision sends a request to the measurement device, which is then responded to with a response frame as shown in Fig. 5. Example of this Simple Request-Response scheme is Get and Set Parameter Request and Response, Get and Set Status Request and Response, and Get and Set Timestamp Request and Response.

ROMULUS Complex Request-Response Scheme: Within the complex Request-Response scheme the communication

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

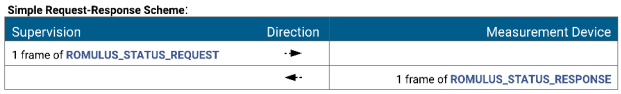


Figure 5: ROMULUS Simple Request-Response Scheme.

could be a Real request-response scheme or a Streaming request-response scheme, as shown in Fig. 6.

The Real request-response scheme communication mode is similar to the simple request-response scheme, with the only difference in being that the measurement device can reply with multiple response frames while the last possible response frame is always fixed to indicate the end of the communication. Example of Real Request-Response scheme is Historical Data Request and Response frames, which initiates communication of multiple frames with the last frame being the Historical Data Done command frame.

In the Streaming request-response scheme, the supervision sends one request to initiate streaming and then the measurement device sends a possibly infinite number of responses until either side breaks the connection or sends a stop command to terminate the stream. Example of Streaming Request-Response scheme is Real-Time Stream request and response which can be terminated by Real-Time Stream Stop command.

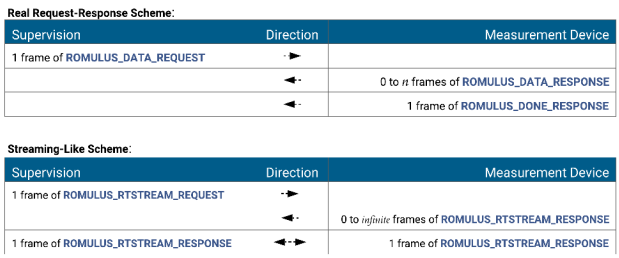


Figure 6: ROMULUS Complex Request-Response Scheme.

Associated with each command are the certain set of user defined parameters and value pair that the user would want to create for their own applications. This is implemented as C Struct data type. Within the structs, the user can define all the different parameters for which a unique ID is automatically generated by ROMULUSlib at compile time which is used in ROMULUS frame construction. Within the C struct, the library currently supports all essential data types: bool, unsigned char/uint8, int32, int64 and float. To categorically define the communicated parameters, eight structs each with a different use case, are defined. A summary of the C struct is provided in Table 1.

ROMULUSlib Architecture and Functions

The data to be communicated is prepared by the CROME devices and handled by ROMULUSlib for communication. ROMULUSlib provides functions for constructing and communicating TCP/IP frames with the supervision system. Frame construction is carried out by functions defined within romulus_frame.h; including function to define the life cycle of a frame, access functions to access and modify specific parts of the frame, validator function for checking the validity of the frame and utility functions to print and debug the frame. Once a valid frame is constructed, functions defined within romulus_net.h are responsible for TCP/IP communication with the supervision system. This includes functions to control and send and receive romulus frames. ROMULUSlib also provides functions to report version and along with run logs, warning and error messages, and also generates timestamps in both, UNIX epoch and human readable form. A brief summary of all important functions of ROMULUSlib with it's placement in the corresponding header is given is Table 2 and a dependency graph of all the ROMULUSlib headers is shown in Fig. 7.

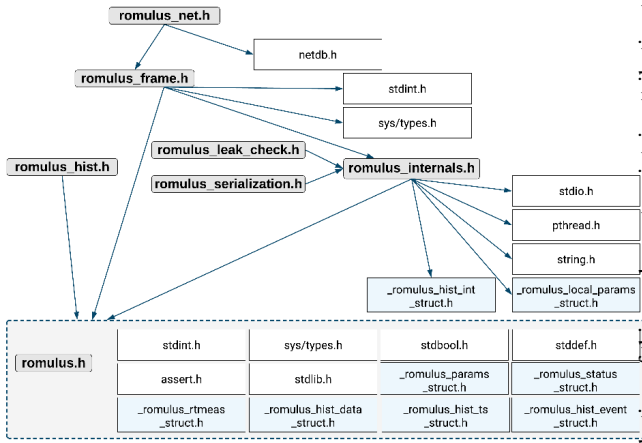


Figure 7: Dependency Graph of all ROMULUSlib Headers.

ROMULUSlib Utilities

ROMULUSlib works as a standalone library for Linux User Space application. In order to facilitate debugging, ROMULUSlib also provides additional utilities. These utilities are meant to aid the developer in writing complete embed-

Table 1: Summary of Supported Data from CROME Organized in C Struct

Struct	Description
romulus_hist_data_struct	Struct for historic data variables.
romulus_hist_event_struct	Struct for hist event variables.
romulus_hist_int_struct	Struct for historic data that is not used by the supervision.
romulus_hist_ts_struct	Struct to define time ranges.
romulus_local_params_struct	Struct for Local parameters to parameterize the system locally.
romulus_params_struct	Struct for Romulus parameters used to parameterize the system and read back inherent parameters from the system.
romulus_rtmeas_struct	Struct which contains real time measurements.
romulus_status_struct	Struct which contains variables to indicate system status .

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 2: Summary of ROMULUSlib Functionalities.

File	Description
romulus.c	Contains: - all the different structs which are used by the ROMULUS library, - all debug functions to query and manipulate the amount of log, - time function definitions to generate timestamp in machine and human readable form, - all function definitions to read the version of the library, - all function and data type definitions for ROMULUSlib's internal types, - definitions to create parameter protection masks to specify the write-protected parameters.
romulus_frame.c	Contains: - all structure definitions including data types and constants related to romulus frames, - all functions required to define the life-cycle of a TCP/IP frame such as header allocation, initialization, allocation, sequence ID generation and free frame memory, - all functions required to access parts of the TCP/IP frame such as, read header, footer, response code and data with offset and append header, footer and data to the frame, - a specific function to create request and response frames, including functions to check validity of the frames, - a specific function to update the parameters of the struct with contents of the TCP/IP frame, - a specific function which are used to manipulate/use status frames, historical data and events frame and real-time measurement frame, - utility functions such as print frame for debugging.
romulus_serialisation.c	Functions to read struct data from files.
romulus_hist.c	All definitions concerning historic data and events.
romulus_internals.c	- Struct, variable & function definition for romulus local parameters, internal historical data. - Definition of printing and IO functions. - Definition of internal data structures. - Struct, variable and function definition for romulus checksum. - Definition of some utility macros.
romulus_leak_check.c	Functions for leack checking, The leakcheck functions print information about memory allocations and calls to free().
romulus_net.c	- All type definitions for romulus frame based network connections. - All functions required to manage the life-cycle of connections to transmit romulus frames - All functions required to send and receive romulus frames. - All functions required to control the sending and reception of romulus frames.

ded applications. Some of these utility applications are as follows:

1. remote_stream: Remote Stream application queries the Embedded Application running on PS and prints the TCP/IP frames for Real-Time Data streaming on port n+1.

2. struct_info_printer: Prints the all Struct Information such as member name, ID and data type as defined by the user.

3. remote_dump: Remote Dump application queries the Embedded Application running on PS and prints status, parameters and hist data by fetching the TCP/IP frames streaming on port n.

RESULTS

At present, 150 CROME devices have been deployed at CERN as the new generation of radiation monitors, all communicating simultaneously with the REMUS supervision via ROMULUSlib ver6.2. The next major release for ROMULUSlib ver7.0 is tested and is scheduled for deployment in December 2021. All tests of ROMULUSlib are performed by a standalone regression testing framework, RomLibEmu[9]. RomLibEmu is developed in Python and works independently of ROMULUSlib to test CROME device's reaction to misconfigurations. This provides a stress testing framework for the application's network interface and test the overall robustness of the software towards the injected faults errors.

EPICS Integration

CROME radiation monitors equipped with ROMULUSlib are currently also in use at the European Spallation Source (ESS) in Lund, Sweden. The ESS' Radiological and Environmental Monitoring System (REMS)[16] uses a two fold approach: Influx DB and Grafana approach for quick online integration, and EPICS integration for real time DAQ and control.

The ESS REMS development team has integrated EPICS [17], the software infrastructure for development

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

of distributed control systems widely used in particle accelerators, large physics experiments and telescopes. The ESS made use of the asynDriver module, a general purpose driver that integrates the device code to the hardware-level code. To integrate EPICS with CROME, initialization files to establish communication link to CROME devices have to be defined using ROMULUSlib. Suitable database files to describe the record types and names of the process variables have to be defined. The CROME devices communicate with the EPICS Input-Output-Controllers that are deployed on a dedicated virtual machine as services and launched at start-up. The collected measurements are posted to an influx database which is then made available via the Grafana interface on the ESS network.

CONCLUSION AND FUTURE WORK

In this paper, we have presented ROMULUSlib, a standalone TCP/IP networking library developed in C. We have presented the ROMULUS data packet structure within the TCP/IP frame, and the communication protocol through request-response mechanism which allows supervision system to request single frame, multiple frames or streaming data frames. Besides this, a comprehensive feature list of all ROMULUSlib functionalities is presented along with debug tools. ROMULUSlib communicates seamlessly with SCADA supervision systems for DAQ and control applications while reliably carrying out millions of data packet transactions every hour (e.g. REMUS supervision handles 2,000,000 measurements/hour from CROME devices currently operational at CERN).

ROMULUSlib makes use of C Struct for storage of application variables of corresponding data types. These C Struct representation allows users to define their own variables as per the application requirement and recompile to autonomously generate ROMULUSlib executable in either arm32 or x86_64 architectures with the new set of application variables. ROMULUSlib currently supports bool, unsigned char/uint8, int32, int64 and float data types to define application variables. Support for uint32 and uint64 will be added in the next versions.

The library has also been developed into a Labview VI using native Labview primitives which allows fast testing of new features and debugging applications. ROMULUSlib currently has been tested for reliability using RomLibEmu, an independent software framework developed in Python; and is presently operational with REMUS supervision system at CERN in Geneva, Switzerland and REMS EPICS supervision system at ESS in Lund, Sweden. In the future versions, we intend to add more regression tests for the library and while robust in application, we would be adding further to the instruction set as per application requirement, along with scaling the library to support varied processor architectures.

ACKNOWLEDGEMENTS

The authors would like to thank Markus Widorski (CERN) for REMUS integration tests, Alasdair Day (ESS) and Juha Hast (ESS) for the performance tests at ESS and the integration of the library into EPICS supervisory system.

REFERENCES

[1] *CROME*, https://crome.web.cern.ch/.
[2] H. Boukabache *et al.*, "Towards a novel modular architecture for cern radiation monitoring," *Radiation protection dosimetry*, vol. 173, no. 1-3, pp. 240–244, 2017.
[3] C. Toner *et al.*, "Fault resilient fpga design for 28 nm zynq system-on-chip based radiation monitoring system at cern," *Microelectronics Reliability*, vol. 100, p. 113 492, 2019.
[4] H. Boukabache, "Crome remote management of soc-based radiation monitors both at cern and ess," System-on-Chip 2nd Workshop - CERN, 2021. https://indico.cern.ch/event/996093/
[5] K. Ceesay-Seitz, H. Boukabache, and D. Perrin, "A functional verification methodology for highly parametrizable, continuously operating safety-critical fpga designs: Applied to the cern radiation monitoring electronics (crome)," in *SAFECOMP 2020 : International Conference on Computer Safety, Reliability, and Security*, 2020.
[6] S. K. Hurst, H. Boukabache, and D. Perrin, "Overview of a complete hardware safety integrity verification according to iec 61508 for the cern next generation of radiation monitoring safety system," in *Proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference*, 2020.
[7] A. Ledeul, A. Savulescu, G. S. Millan, and B. Styczen, *Data streaming with apache kafka for cern supervision, control and data acquisition system for radiation and environmental protection*, 2019.
[8] A. R. Ledeul, "Integration of crome into remus," SoC Interest Group Meeting, 2020. https://indico.cern.ch/event/882283/
[9] K. Ceesay-Seitz, M. Leveneur, H. Boukabache, and D. Perrin, "Romlibemu: Network interface stress tests for the cern radiation monitoring electronics (crome)," in *18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, 2021.
[10] R. Mastyna, J. Casas-Cubillos, E. Blanco Vinuela, N. Trikoupis, and M. Felser, "Profinet communication card for the cern cryogenics crate electronics instrumentation," 2017.
[11] H. Milcent, E. Blanco, F. Bernard, and P. Gayet, "Unicos: An open framework," in *12th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systems. Grenoble (France)*, 2009, pp. 12–16.
[12] S. Schlenker, C.-V. Soare, D. Abalo Miron, V. Filimonov, B. Farnham, and P. Nikiel, "Quasar-a generic framework for rapid development of opc ua servers," 2015.
[13] M. G. D. Gololo, "Soc developments for the detector control system of atlas tile calorimeter at the hl-lhc," System-on-Chip 2nd Workshop - CERN, 2021. https://indico.cern.ch/event/996093/
[14] E. Smith, "Zynq us+ mpsoc in the gfex hardware trigger in atlas," System-on-Chip 2nd Workshop - CERN, 2021. https://indico.cern.ch/event/996093/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR01

[15] P. Moschovakos, "Socs for detector controls and their applications," System-on-Chip 2nd Workshop - CERN, 2021. https://indico.cern.ch/event/996093/

[16] J. Hast, "Ess use case of the crome monitor with epics," 2nd System-on-Chip Workshop - CERN, 2021. https://indico.cern.ch/event/996093/

[17] *EPICS*, https://epics-controls.org/.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

CONTROL, READOUT AND MONITORING FOR THE MEDIUM-SIZED TELESOPES IN THE CHERENKOV TELESCOPE ARRAY

U. Schwanke*[1], T. Murach[2], P. Wagner†[2], G. Spengler[1]
for the CTA MST Project and
D. Melkumyan[2], I. Oya[3] and T. Schmidt[2]
[1] Humboldt-University, Berlin, Germany
[2] DESY, Zeuthen, Germany
[3] CTA, Heidelberg, Germany

Abstract

The Cherenkov Telescope Array (CTA) is the next-generation ground-based gamma-ray observatory. Its design comprises several ten imaging atmospheric Cherenkov telescopes deployed at two sites in the southern and northern hemisphere. The inclusion of various array elements, like large-sized, medium-sized and small-sized telescopes, instruments for atmosphere monitoring, etc, into the Array Control and Data Acquisition System (ACADA) poses a particular challenge which is met by an appropriate software architecture and a well-defined interface for array elements. This paper describes exemplarily how the interface is implemented for the Medium-Sized Telescopes (MSTs, 12 m diameter). The implementation uses the ALMA Common Software (ACS) as a framework for software applications facilitating the readout and control of telescope subsystems like the drive system or the pointing camera; the communication with subsystems takes advantage of the OPC UA protocol.

INTRODUCTION AND OVERVIEW

The Cherenkov Telescope Array (CTA) is the next-generation ground-based observatory for gamma-rays in the energy band between some 10 GeV and several 100 TeV. CTA will comprise two arrays of Imaging Atmospheric Cherenkov Telescopes (IACTs) located in the southern (Paranal, Chile) and northern (La Palma, Canary Islands, Spain) hemisphere, respectively. IACTs with different mirror areas (dubbed Large-Sized Telescopes (LSTs), Medium-Sized Telescopes (MSTs) and Small-Sized Telescopes (SSTs)) ensure proper detection efficiencies for gamma-rays over four orders of magnitude in energy. The stereoscopic observation of showers with numerous IACTs results in an angular resolution and flux sensitivity that constitutes a substantial improvement compared to current IACT arrays (like H.E.S.S., MAGIC, and VERITAS). In CTA's (initial) α-configuration, the northern installation will comprise 4 LSTs and 9 MSTs; 14 MSTs and 37 SSTs will be installed at the southern site.

The readout, control and monitoring of different array elements (telescopes and atmospheric monitoring devices) is the task of the Array Control and Data Acquisition (ACADA) system. The ACADA system is quite complex since it also needs to cope with the concurrent automatic operation of multiple IACT sub-arrays and the rapid re-scheduling of observations in response to science alerts generated internally or by external astronomical facilities. The ACADA software is based on an architecture designed using the Unified Modeling Language (UML) and Systems Modeling (SysML) formalisms [1]. The implementation of applications takes advantage of the ALMA Common Software (ACS [2]) framework.

It adds to the challenge for the ACADA system that the three telescope types (LSTs, MSTs, SSTs) use different optics designs, employ different auxiliary hardware devices (e.g. reference lasers, pointing cameras, light flashers) and implement different operational procedures (observations, calibration, monitoring). A key assumption of the software design is that ACADA can control any telescope without knowing its type and that all telescopes implement a common stateful behaviour via a finite state machine (FSM). This paper describes the software aspects of the interface between ACADA and a telescope. It uses the MSTs as the primary example and starts therefore with a description of the telescope hardware. The implementation of the software interface using ACS and the entire MST software system are detailed as well.

THE MSTS FOR CTA

Telescope Optics

The MSTs [3] are IACTs with a field of view (FoV) of about 8° in diameter and an effective mirror area of about 88 m^2. In the CTA installations, they will be arranged at a typical distance of $O(100)$ m from each other and will cover the central energy range between 100 GeV and 30 TeV. The tesselated mirrors consist of 86 hexagonal spherical mirror facets with a focal length of $f = 16.07$ m that are arranged on a sphere with a radius of curvature of $R = 19.2$ m. The telescope focal length in this so-called modified Davies-Cotton design is $F = 16$ m. The heart of the telescope is a camera[1] with a flat surface that is located in the telescope focal plane and uses about 1800 PMT pixels for the detection of Cherenkov light.

There are two Cherenkov camera projects differing in the number of pixels and the way of storing and processing PMT

* schwanke@physik.hu-berlin.de
† Software for Science (Berlin)

[1] In the following, the term *camera* without further qualification is used only for the Cherenkov camera.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

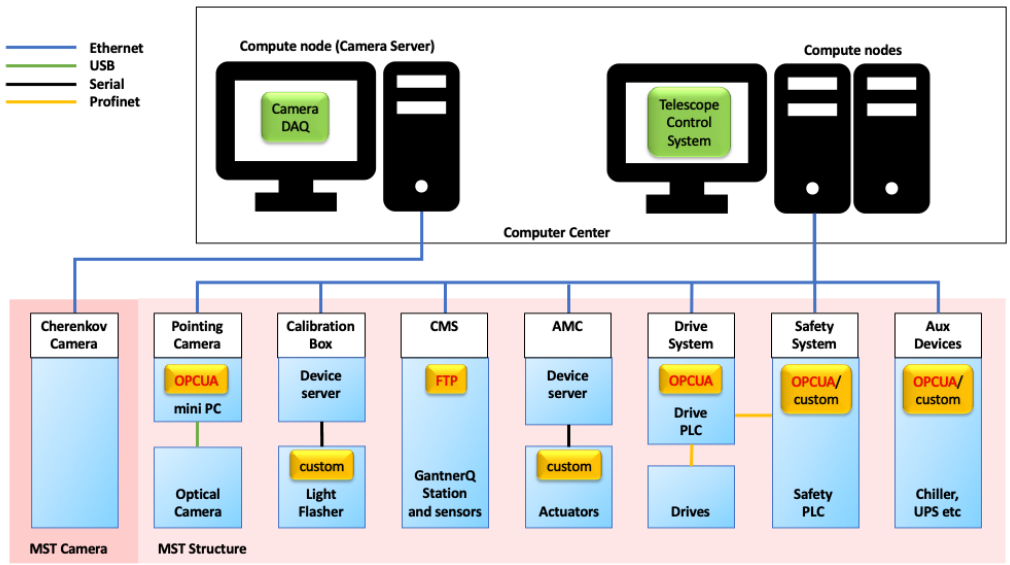


Figure 1: Breakdown of hardware and software systems involved in the operation of an MST. Rectangles denote hardware units. Protocols are denoted by yellow rectangles with rounded corners. See text for explanations.

signals. The MSTs in the southern CTA installation will be equipped with FlashCam cameras (1764 pixels, FADC-based data acquisition system [4]); NectarCAM cameras (1855 pixels, temporary storage of signals in an analogue ring buffer [5]) will be deployed in the north. It is beyond the scope of this paper to do justice to both camera projects. In the following, the focus will therefore be on the *MST structure* (everything but the Cherenkov camera). The description of the camera will be somewhat coarser and will only address those conceptual properties common to FlashCam and NectarCAM.

Telescope Subsystems

The Cherenkov camera is the most important subsystem of an MST. The acquisition of camera data will result in data streams of about 1 GB/s per MST. Next to the camera there are the following subsystems that have to be handled by the telescope control software (cf. lower part of Fig. 1):

- **Pointing Camera:** The position of the Cherenkov camera with respect to the MST structure and stars in the sky is measured by a wide-FoV (about $15° \times 20°$) optical camera. The pointing camera is located in the centre of the telescope mirror and roughly aligned with the optical axis of the MST. It images simultaneously stars in the sky and positional LEDs at the camera.
- **Calibration Box:** A light flasher[2] is mounted next to the pointing camera. It illuminates the Cherenkov camera with the aim of recording the response of the camera pixels to short light flashes. Equalizing the pixel response is part of the calibration of the Cherenkov camera.
- **Conditions Monitoring System (CMS):** Preventive maintenance of the telescope hardware shall be accomplished by the usage of a CMS [6] comprising $O(10)$ acceleration and displacement sensors along with a dedicated data acquisition system for the sensor readings. The sensors are mounted at components of the drive system (motors, gear units, worm shaft) and at locations in the telescope structure. Time-dependent changes of the sensor response to external (e.g. wind) or internal (e.g. well-defined telescope movements) excitations would be indicative of problems.
- **Automatic Mirror Control (AMC):** The 86 mirror facets have to be aligned such that light from a point source is imaged into a small spot in the telescope focal plane. For this purpose, the mirror facets have a triangular fixation, and two fixation points of each facet can be moved with the help of stepper motors (actuators) by a few centimeters. The proper actuator settings are typically found in a calibration process using images of stars in the telescope focal plane recorded by the pointing camera [7].
- **Drive System:** The MSTs have an azimuth-elevation mount. Telescope movements are facilitated by four elevation ($-20° < \text{el} < 91°$) and two azimuth ($-270° < \text{az} < 270°$) motors moving the corresponding drives based on encoders readings.
- **Safety System:** Safety of humans and equipment is ensured by a number of sensors, hardware switches (e.g. emergency buttons) and interlocks. The corresponding signals are captured and processed by a safety system which also communicates with the other subsystems.
- **Auxiliary Devices:** There are a number of smaller subsystems (e.g. a chiller for the cooling of the Cherenkov

[2] The description here and in the following refers to MSTs equipped with NectarCAMs.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

camera and the telescope structure, an uninterruptible power supply (UPS)) that produce slowly varying sensor readings. The measurements comprise temperatures, pressures, humidities and rates of flow.

The drive system and the Cherenkov camera are vital for scientific observations. A failure of the pointing camera during observations would result in data sets with reduced pointing accuracy. It is expected that the mirror alignment obtained with the help of the AMC and the pointing camera is so stable that AMC only needs to be operated during the alignment or in emergency situations (e.g. to defocus the telescope to prevent damage due to the Sun). A CMS will be deployed and studied at the first MST scheduled for construction at the northern CTA site.

THE ACADA–TELESCOPE INTERFACE

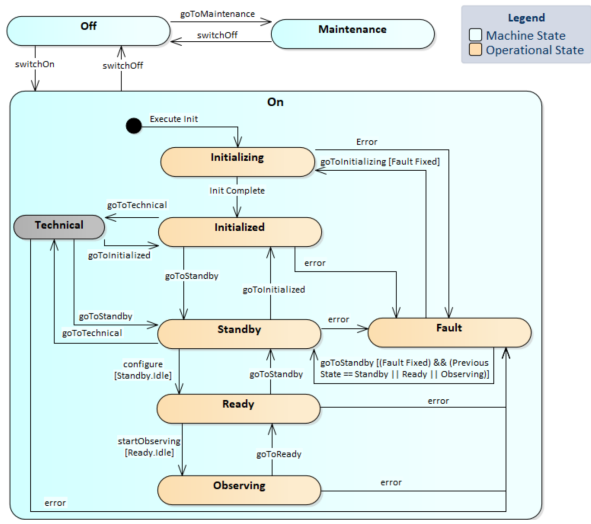


Figure 2: State machine of a CTA telescope. There are three machine states, of which the On state has several operational sub-states. Transitions between states are indicated by arrows.

The generic interface for CTA telescopes is primarily used by ACADA. This interface describes high-level methods that are primarily used to control telescopes (and other types of array elements, which will be omitted in the discussion below) and to obtain basic information about the state of the respective array element. The most important set of methods defined in the interface relate to state transitions of the array element. An incomplete list of relevant transitions is shown in Table 1. These transitions, together with the states and sub-states a telescope can assume, are also visible in Fig. 2. As can be seen from the table, these are high-level methods, and detailed control of e.g. individual hardware elements is not intended. The usage of configuration databases is assumed and reflected in certain interface method signatures. For example, both the telescope structure and the camera can load different configurations depending on the exact type of observations that shall be conducted. The respective

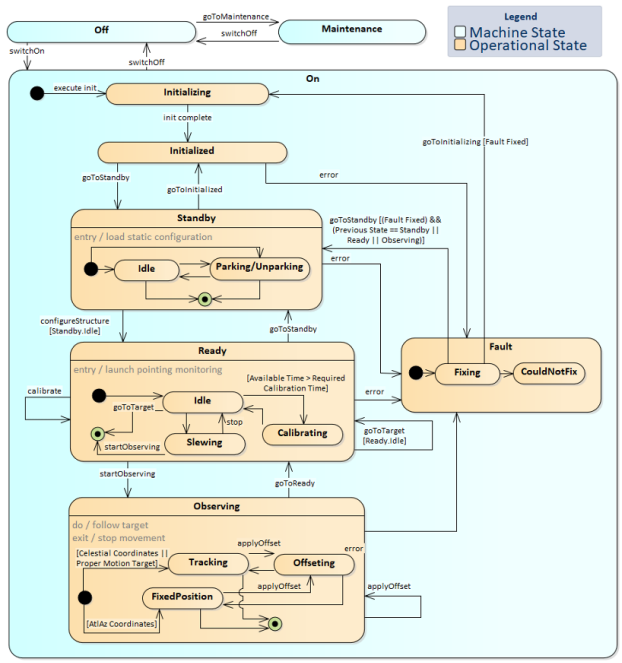


Figure 3: State machine of the structure of a CTA telescope. There are three machine states, of which the On state has several operational sub-states. Four of the operational states have sub-states themselves. Transitions between states are indicated by arrows.

Table 1: Selection of the most relevant telescope transitions. These transitions represent the core of the telescope interface. The last three transitions can optionally start observations with the camera. See Fig. 2 for a list of telescope states and allowed transitions.

Transition	Description
goToInitialized goToStandby goToReady goToTechnical	Puts telescope to the Initialized/Standby/ Ready/Technical state
startObserving	Starts observation of a previously defined target
goToSkyTarget	Lets telescope track a celestial coordinate
goToProperMotionTarget	Lets telescope follow a target by providing a trajectory
goToFixedPosition	Points telescope to a fixed position

interface method passes configuration IDs to a so-called TelescopeManager (cf. upper part of Fig. 4) which is the highest-level abstraction of a telescope. These IDs are then used to look up configuration parameters from a database.

For standard observations, the interface provides only methods for controlling the TelescopeManager. Access to subsystems, like the telescope structure or the camera, is not foreseen. For special-purpose operations the telescope

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

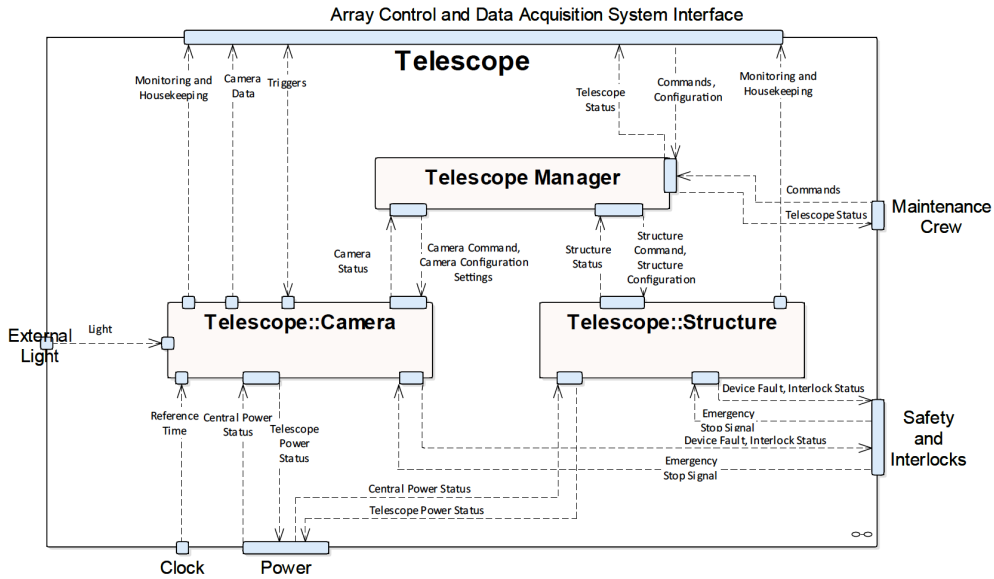


Figure 4: High-level view of the interfaces of the TelescopeManager, the camera and structure subsystems. When performing standard observations, ACADA only interacts with the TelescopeManager, which is in charge of supervising and controlling subsystems. When the telescope is in the Technical state, ACADA accesses and interacts with the two telescope subsystems directly. External devices and actors interacting with the telescope manager and subsystems are displayed as well. The exchange of commands and data is displayed through arrows.

can enter the Technical state. In this state, ACADA can request from the TelescopeManager access to an extended interface with direct control over the Cherenkov camera and the structure (CameraManager and StructureManager in Fig. 4). Also more detailed information about the telescope and subsystem statuses is available in this mode.

When maintenance work is ongoing at a telescope, the TelescopeManager shall reflect this by assuming the Maintenance state. The respective transition is initiated by a physical switch at the telescope. In this state, any remote control by e.g. ACADA is prohibited to ensure human safety. Obtaining monitoring data is still possible in this state, though.

The monitoring of telescope parameters and conditions is not part of the interface. Also the transfer of bulk data from the cameras is not specified through the interface discussed here.

The telescope interface is implemented from both sides. For the development of the user side, i.e. the ACADA system, a mock implementation of a generic telescope has been developed in Python. Telescope states, relevant transitions and subsystem transitions have been implemented to mimick realistic and configurable behaviour of telescopes. Threads are used to simulate long-lasting procedures and transitions, and callbacks are used as primary concept for notifying clients of the mock TelescopeManager about the progress of such operations. The mock implementation of the TelescopeManager is used in a variety of automated software tests of ACADA, such as verifications of high-level ACADA use cases.

The interfaces between the TelescopeManager or ACADA on the one side and the StructureManager and the CameraManager on the other side are also defined. As indicated above, ACADA can only access these components when the telescope is in the Technical state.

As shown in Fig. 3, a state machine has also been defined for the structures of CTA telescopes. The TelescopeManager (or ACADA) requests transitions between operational states. Internal states as well as transitions between them are handled by the StructureManager. Sub-states only need to be implemented if applicable.

Several states and sub-states can be entered by the telescope (or subsystem) autonomously. For example, calibration routines can be performed automatically by the telescopes if the telescope was instructed by ACADA to start observing at a given time and there is enough time in advance to calibrate subsystems. In this case, the StructureManager can decide to enter and leave the Ready.Calibrating state. Also in case an error occurred the TelescopeManager can attempt to fix the problem, and if successful, the telescope (or the respective subsystem) can return to a non-Fault state by itself.

READOUT, CONTROL AND MONITORING OF MSTS

Protocols, Standards and Data Formats

The data stream from the camera is dominating the bandwidth that the ethernet network (blue lines in Fig. 1) between an MST and the local computer centre on a CTA site has to provide. In the event of a camera trigger, the digitised PMT signals are transferred to a dedicated computer (*camera server* in the upper left part of Fig. 1) which caches all

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

data for a period of about 1 s and processes the data further. Typically only events where at least one other telescope recorded useable camera data simultaneously are kept and stored on hard disk for offline analysis. The stereoscopy requirement (more than one telescope with data) is enforced by a central software array trigger (SWAT). The SWAT operates on trigger time stamps sent to it by all cameras in the array and informs the camera servers about data of interest. The transfer of camera data uses a highly-performant bulk-data transfer service that is based on the `ZeroMQ` messaging library and can also compress data [8].

All other data streams for readout, control and monitoring are fairly modest in size. The high number of different telescope subsystems and the need to upgrade their hardware and software over the anticipated lifetime of CTA (about 30 years) motivate, however, the prescription of standard protocols and mechanisms for the data transfer. For CTA, the combination of an ACS DeviceIO[3] with the OPC UA industrial communication protocol is the method of choice. OPC UA has been standardised by the OPC Foundation. It has growing support in industry and a number of software development kits exist. An implementation of an ACS DeviceIO for OPC UA has been made available to the telescope projects. To be practical, the following three options for the inclusion of a hardware device/telescope subsystem exist:

1. Devices that come with a vendor-side OPC UA server can be immediately integrated by means of the ACS DeviceIO for OPC UA.
2. For devices that communicate via a dedicated device protocal an OPC UA server needs to be developed. The server will translate between the device protocol and OPC UA and will be integrated as described above.
3. If the development of an OPC UA server is not an option a device with a dedicated protocol can be included by providing an ACS DeviceIO for this protocol.

Local Control Systems of the MST Structure

The usage of ACS is confined to the Telescope Control System (TCS, cf. upper right part of Fig. 1). The TCS provides the higher-level control and implements the telescope interface described above. The division of labour between the TCS and the software systems located at the telescope is motivated by safety aspects and the wish to minimize future changes of the software at the telescope. The communication between the TCS in the computer centre and an MST structure uses ethernet. The chosen hardware and software solutions are as follows (cf. lower part of Fig. 1):

- **Pointing Camera:** The pointing camera consists of an optical camera and a mini PC in a custom-built housing. The housing provides temperature stabilization and features a number of internal sensors. The camera and the mini PC are connected by USB; the mini PC has an OPC UA server for control and image readout.
- **Calibration Box:** The LED-based light flasher is managed by firmware on an 8bit microcontroller using a custom protocol. A device server is used to convert the serial communication to ethernet.
- **Conditions Monitoring System:** The data acquisition for the conditions montoring system uses a real-time board (Gantner Q.station) that digitises the arriving analogue sensor signals and buffers them. At present, the board sends the data to an FTP server in the computer centre. There are also ways to transmit data for storage.
- **Automatic Mirror Control:** The communication with the $2 \times 86 = 172$ actuators uses serial lines and a custom protocol. The serial lines are converted to ethernet with the help of an device server. Each actuator provides not only the current position of the stepper motor but also monitoring information (e.g. a temperature). There is the option to choose between several actuator settings stored in the firmware.
- **Drive System:** The drive system at the telescope needs no knowledge about astronomy and basically only deals with time, azimuth and elevation. A programmable logic controller (PLC) accepts precalculated tables with azimuth and elevations as a function of time in discrete time steps. The PLC code interpolates the information in the table and controls the drive motors such that the intended path in the sky is followed (e.g. to implement siderial tracking). The interface to the outside world is an OPC UA server.
- **Safety System:** The safety system is also a PLC that bundles and processes all safety-relevant information. The communication with the outside world proceeds via an OPC UA server. The Safety PLC offers some extra monitoring data (e.g. tables with motor torques at rates much higher than a few Hz) via a custom protocol. All PLCs in the system (Drive PLC, Safety PLC) are connected via the Profinet bus.
- **Auxiliary Devices:** The readout and monitoring of chiller and UPS will depend on the exact models that will be chosen in the future. It is envisaged that the information (e.g. sensor readings or status flags) will be made available as an OPC UA server.

THE TELESCOPE CONTROL SYSTEM

Structure of the TCS

For the implementation of an MST-specific TelescopeManager, the Java programming language has been chosen. All relevant interface methods have been implemented, and their functionality is verified by a set of unit and component tests. Focus is put on achieving a good test coverage. In the following, only the StructureManager will be discussed in detail since the camera software design depends on the deployed hardware (FlashCam or NectarCAM).

For the MST StructureManager, the common interface has been extended by a set of custom methods that are required to control those subsystems (e.g. the AMC and the

[3] An ACS DeviceIO provides a mapping between a node in a CORBA object (e.g. Property in an ACS Component) and a monitoring/control point in an external hardware device.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

pointing camera) that are only available for MSTs. ACADA is not aware of these methods, but the TelescopeManager makes use of them for both commissioning and regular operations. The extended interface of the StructureManager is described by an appropriate IDL[4] file. Placing the extended interface in the StructureManager has the advantage that the full functionality of an MST structure is available to clients. This removes the need for clients to interact directly with subsystems controlled by the StructureManager.

Implementation as ACS Components

The TCS for an MST is implemented as a flat hierarchy of ACS components. The TelescopeManager (cf. Fig. 5) controls the CameraManager and StuctureManager and calculates its state as a function of the states of the two subsystems. The StructureManager interacts directly with the Safety System to interrogate the hardware state (interlocks, emergency switches) of the MST structure and includes this information in the computation of its own state. There are four basic groups of ACS components that aid the StructureManager to read out and control the MST structure hardware:

- **Pointing Camera:** The top-level ACS component for the pointing camera receives configuration information (rates and exposures for image acquisition) and launches threads for the acquisition of images from the pointing camera. A *Bridge* ACS component is used for mapping to the OPC UA protocol.
- **Automatic Mirror Control:** The top-level ACS component for the AMC deals with the AMC configuration (number and arrangement of actuators) and accesses an OPC UA server via a *Bridge*. The OPC UA server for the AMC is executed locally and uses a custom protocol for the actual communcation with the hardware.
- **Drive System:** The ACS component for the drive system forwards simple commands (e.g. pointing the telescope to a fixed position in azimuth and elevation) directly to the drives system hardware. For tracking of astronomical targets it generates the time-dependent trajectory in azimuth and elevation in time steps of about 250 ms. Pointing corrections[5] are applied through a a pointing model in this step. About 1000 trajectory steps (corresponding to 250 seconds of tracking) are initially sent to the drive PLC and buffered there. The buffer is topped off whenever necessary.
- **Auxiliary Devices:** The readout and control of the auxiliary devices is managed by ACS components coupled to a *Bridge*.

Clients and Graphical User Interfaces

ACADA is the most important user of the MST TCS. It regularly inspects the TelescopeManager state to check the availability of the TCS (and hence the telescope hardware) for standard obervations. For a typical night with observations ACADA prepares the telescopes well before dark time. In this preparatory phase, the telescopes leave the parking position and execute telescope-specific routines (attain the operating temperature for the Cherenkov camera, record calibration data). Observations of scheduled astronomical targets are conducted when the conditions are optimal (astronomical darkness, partial Moon). There is also a closing phase near dawn; it is complementary to the preparatory phase and ends with the telescopes returning to the parking positions.

In periods of telescope commissioning, extended calibration and trouble-shooting, expert personnel requires a more fine-grained access to the TCS. For this purpose, the TelescopeManager can be brought to the *Technical State* (cf. Fig. 2) and signals in this way that a remote control by ACADA is prohibited. There a then two mutually exclusive ways to control an MST: by means of a graphical user interface (GUI) and by means of scripts that manipulate the TCS directly. In both approaches, the TelescopeManager must finally leave the *Technical State* again so that ACADA can reclaim control.

Figure 6 shows a screenshot of the browser-GUI that can be used for the MST structure. The underlying server takes advantage of the same software technologies that are used for the CTA operator GUI. The server is written in Python as an application of the `Pyramid`[6] open-source web framework. It uses native Python features (`asyncio`) and the websockets library to implement concurrency and the communication with a client, respectively. Internally, the server gains access to ACS components of the TCS. The offered functionality comprises actions with the drive system (track astronomical targets, point the MST to a coordinate in azimuth and elevation etc) and basic tests with the AMC (actuator movements). A passive monitoring of basic telescope parameters (e.g. the current pointing direction, cf. Fig. 6) is available at all times while active control of the MST structure is only possible in the *Technical State*.

Client scripts (in Python) are the most flexible way to access the TCS. While ACADA works exclusively with the TelescopeManager the scripts can also approach the StructureManager, the CameraManager or any of the underlying ACS components. A typical example for the MST structure is a script implementing a procedure for mirror alignment. In this application, the script would use the standard interface of the TelescopeManager to initiate the tracking of a star. The extended interface of the StructureManager is then used to vary the orientation of the mirror facets (by means of the actuators) and to record images with the pointing camera.

ACKNOWLEDGEMENTS

The authors acknowledge the work of numerous members of the CTA consortium in the ACADA and MST Projects which helped to shape the MST and ACADA software. We gratefully acknowledge financial support from the agencies

[4] interface definition language

[5] Pointing corrections are needed whenever the azimuth and elevation of the drive system are not identical with the astronomical azimuth and elevation.

[6] https://www.trypyramid.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR02

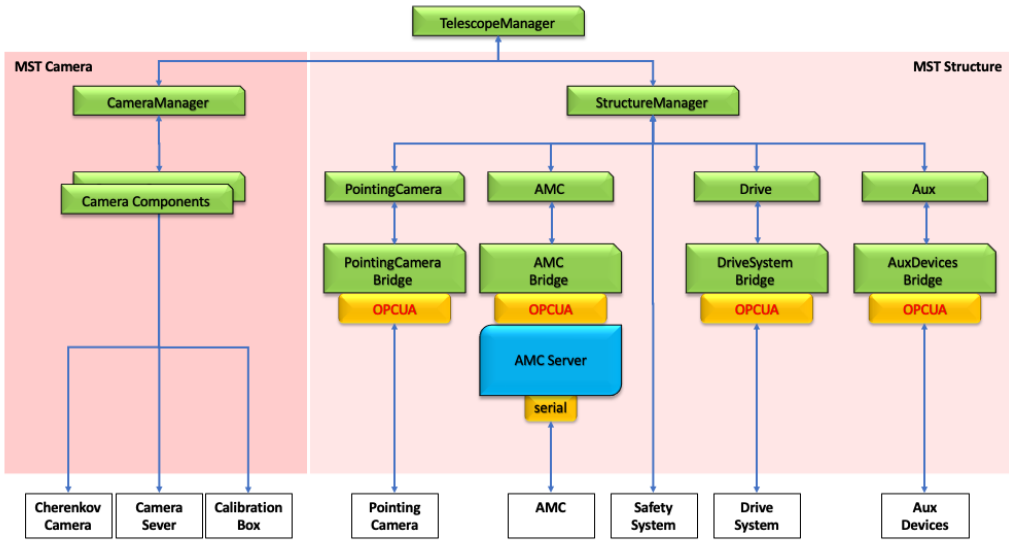


Figure 5: Breakdown of the TCS executed for each MST in the computer centre. Green boxes denote ACS components; the blue box with two rounded corners is an OPC UA server that runs locally. Arrows indicate both control and data transfer. The hardware at the MST that is accessed by the TCS is shown by the boxes at the bottom. Protocols are denoted by rectangles with rounded corners. The *Bridge* ACS components provide the mapping to OPC UA. Note that the number of ACS components for the Cherenkov camera depends on the camera project. See text for more explanations.

Figure 6: Screenshot of the browser-based GUI for monitoring and control of the MST structure. The black line illustrates the trajectory of the telescope pointing direction in azimuth and elevation.

and organizations listed here:
http://www.cta-observatory.org/consortium_acknowledgments

REFERENCES

[1] I. Oya *et al.*, "Designing and prototyping the control system for the cherenkov telescope array," *Journal of Physics: Conference Series*, vol. 1085, p. 032 045, 2018, doi:10.1088/1742-6596/1085/3/032045

[2] G. Chiozzi *et al.*, "The ALMA common software: a developer-friendly CORBA-based framework," in *Advanced Software, Control, and Communication Systems for Astronomy*, vol. 5496, 2004, pp. 205–218, doi:10.1117/12.551943

[3] A. Schulz, M. Garczarczyk, L. Oakes, S. Schlenstedt, and U. Schwanke, "Building medium size telescope structures for the Cherenkov telescope array," in *6th International Symposium on High Energy Gamma-Ray Astronomy*, vol. 1792, 2017, paper 080008, p. 080 008, doi:10.1063/1.4969029

[4] B. Bi *et al.*, "Performance of the New FlashCam-based Camera in the 28 m Telescope of H.E.S.S," *arXiv e-prints*, paper arXiv:2108.03046, arXiv:2108.03046, 2021.

[5] T. Tavernier, J. F. Glicenstein, and F. Brun, "Status and performance results from NectarCam - a camera for CTA medium sized telescopes," in *36th International Cosmic Ray Conference (ICRC2019)*, vol. 36, 2019, paper 805, p. 805.

[6] V. Barbosa Martins, G. Spengler, M. Garczarczyk, U. Schwanke, MST-STR Project, and CTA Consortium, "A Condition Monitoring Concept Studied at the MST Prototype for the Cherenkov Telescope Array," in *36th International Cosmic Ray Conference (ICRC2019)*, vol. 36, 2019, paper 626, p. 626.

[7] T. Murach *et al.*, "Automatic mirror alignment for the medium-sized telescopes of the Cherenkov Telescope Array using the Bokeh method," in *Ground-based and Airborne Telescopes VII*, vol. 10700, 2018, paper 107001U, 107001U, doi:10.1117/12.2313517

[8] E. Lyard, R. Walter, and C. Consortium, "End-to-end data acquisition pipeline for the Cherenkov Telescope Array," in *35th International Cosmic Ray Conference (ICRC2017)*, vol. 301, 2017, paper 843, p. 843.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

HEXAPOD CONTROL SYSTEM DEVELOPMENT TOWARDS ARBITRARY TRAJECTORIES SCANS AT SIRIUS/LNLS

A. Y. Horita *, F. A. Del Nero, M. A. L. Moraes, G. N. Kontogiorgos, LNLS, Campinas, Brazil
G. G. Silva, UNICAMP, Campinas, Brazil

Modern 4th generation synchrotron facilities demand high precision and dynamic manipulation systems capable of fine position control, aiming to improve the resolution and perfomance of their experiments. In this context, hexapods are widely used to obtain a flexible and accurate 6 Degrees of Freedom (DoF) positioning system, since it is based on Parallel Kinematic Mechanisms (PKM). Aiming the customization and governability of this type of motion control system, a software application was entirely modeled and implemented at Sirius. A Bestec hexapod was used and the control logic was embedded into an Omron Delta Tau Power Brick towards the standardization of Sirius control solutions with features which completely fill the beamline scan needs, e.g., tracing arbitrary trajectories. Newton-Raphson numerical method was applied to implement the PKM. Besides, the kinematics was implemented in C language, targeting a better runtime performance when comparing to script languages. This paper describes the design and implementation methods used in this control application development and presents its resulting performance.

INTRODUCTION

Sirius is the 4th generation synchrotron light source being commissioned by the Brazilian Synchrotron Light Laboratory (LNLS), in Brazil [1]. Its low emittance (0.25 nm rad) makes it one of the world's brightest light sources of its kind [2]. In this sense, high precision control systems are required towards better beam quality and stability.

In Sirius IPE Beamline, a Bestec P468 Mirror Unit [3] was chosen as the motion system of a toroidal mirror. This unit contains a hexapod in its structure, which is illustrated in Fig. 1. Hexapods are widely used due to the pararell control capacity of its 6 degree of freedom (DoF), characterized in Pararell Kinematics Machines (PKM) [4].

Together with this unit, a Bestec P494 Motion Control System was also acquired, which embeds a Mint Linux ditribution as Operating System (OS), containing a set of proprietary control software which implements the user interface, configuration files, system kinematics and the hexapod system interface.

Although the P494 control system meets the beamline expected stability and motion control accuracy, a customized system developed in-house was desired towards Sirius's systems standardization and independence when implementing new features, such as arbitrary trajectories scans. Under these circumstances, we decided to implement the control logics using an Omron Delta Tau Power Brick LV controller [5].

* augusto.horita@lnls.br

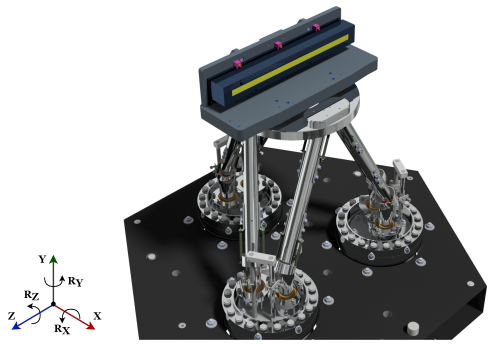


Figure 1: Toroidal mirror supported by hexapod in the Bestec P468 Mirror Unit.

An important portion of the motion control logics consists of its kinematics, which is splitted into two algorithms.The inverse kinematics converts system positions to individual motors' positions and the forward kinematics calculates the system feedback positions based on encoders [6]. Towards a more productive and robust implementation process, the hexapod kinematics was first mathematically modeled using a Jupyter Notebook [7], which provided flexibility using software structures, e.g. mathematical libraries, and optimized the simulation and bugfixes process [8]. Due to the fact that the kinematics of a hexapod is non-linear [6], our model was based on Newton Rhapson for root-finding, as it is widely used and has a fast convergence rate [9].

Based on the validated Jupyter Notebook model, it was possible to implement the control logics in the Delta Tau Power Brick LV system. Its performance was then compared to Bestec's control sytem for validation. Also, an example of arbitrary trajectory was implemented for functionality validation purposes.

The remaining sections of this paper presents in more details the used methods for this control system development and its validation.

MOTION CONTROL DEVELOPMENT

In this section, we present the motion control system development, focusing on the kinematics modeling and implementation.

Kinematics Model

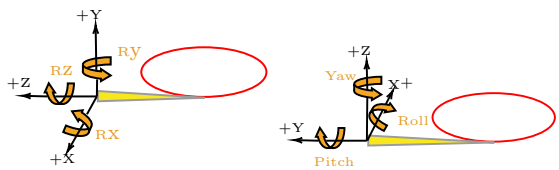
In the presented application, the toroidal mirror is placed inside a vacuum chamber. Towards a more robust and easy

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

for maintenance architecture, Bestec's P468 Mirror Unit was designed placing the actuators outside the vacuum environment. In this context, this Mirror Unit hexapod design slightly differs from the usual hexapod configuration found in other applications and in literature [6], as illustrated in Fig. 1. In this design, the six manipulator's struts have constant length, and the platform movements are caused by the vertical displacement of six linear actuators placed outside the chamber, and connected to the struts by a spherical joint. As a consequence, the presented kinematics model also differs from models found in the literature.

For standardization purposes, the kinematics model and the mathematical formulation presented is this work are based on the Sirius coordinate system, in which the *Z*-axis points to beamlight direction, *X*-axis outward from the storage ring and *Y*-axis is pointed upwards, as shown in Fig. 2a. The Bestec motion control sytem is based on a different coordinate system, as illustrated in Fig. 2b. This difference is compensated during the data analysis by comparing the correspondent DoF of each control system. Aiming the simplification of methods description, the remaining of this paper refers to the Sirius coordinate system.



(a) Sirius (b) Bestec

Figure 2: Coordinate system conventions

Two auxiliary frames are used to model the kinematics transformations: the platform's coordinate system $O_p(u, v, w)$ (at platform centroid) and the granite plate coordinate system $O_g(x, y, z)$. In addition, a set of vectors is also necessary, as illustrated in Fig. 3. Let $\vec{b}_i$ (i ∈ 1, …, 6) be the vectors from O_g to the ith granite joint. Similarly, let $\vec{a}_i$ (i ∈ 1, …, 6) be the vectors from O_p to the ith platform joint, written with respect to platform's coordinate system. Moreover, let $\vec{l}_i$ be the vector representing the ith hexapod strut, with ($\|\vec{l}_i\|$ being constant. The $\vec{s}_i$ links the ith granite joint to the ith platform joint, and $\vec{c}_i$ the vertical vector representing the position of the ith linear actuator, in reference to the granite plate. Finally, let $\vec{p}$ be the position vector of the platform centroid O_p in reference to O_g and $\vec{m}$ the position vector of the end-effector *i.e.*, the mirror centre, considering the O_g coordinate system.

We also define a rotation matrix R as the product of three consecutive rotations along each coordinate axis *i.e.*, $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$, considering that:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad (1)$$

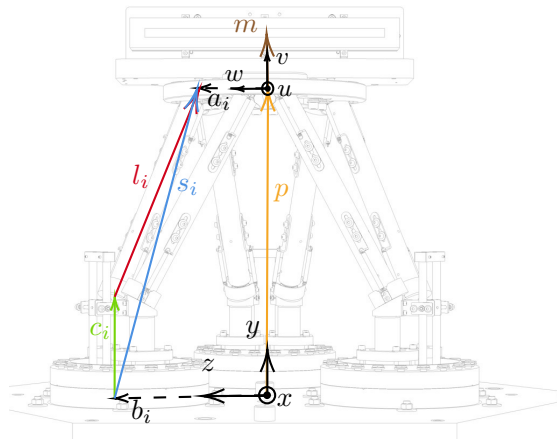


Figure 3: Coordinate systems and vectors applied to hexapod model.

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad (2)$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Inverse Kinematics Model Given a position $(x, y, z, \alpha, \beta, \gamma)$ of the mirror centre, we present the algorithm to find each linear actuator displacement. Based on Fig. 3, and considering that the platform and the mirror are coupled, their angular position (α, β, γ) are the same. In this sense, it is possible to find the platform's centroid O_p position, represented by vector $\vec{p}$, by subtracting the vector $\vec{m}$, of fixed module $\|\vec{m}\|$, from mirror centre position (x, y, z). Figure 3 also illustrates that vector $\vec{s}$ can be calculated as shown in Eqs. (4) and (5):

$$\vec{s}_i = \vec{p} + R \cdot \vec{a}_i - \vec{b}_i \quad (4)$$

$$\vec{s}_i = \vec{c}_i + \vec{l}_i \quad (5)$$

Considering $\vec{c}_i = (0, y_{ci}, 0)$, we can write:

$$\vec{s}_i - \vec{c}_i = (x_s, y_s - y_c, z_s) \quad (6)$$

As the hexapod struts lengths $\|\vec{l}_i\| = \|\vec{s}_i - \vec{c}_i\|$ are constant:

$$y_{ci} = y_{si} - \sqrt{\|\vec{l}_i\|^2 - z_{si}^2 - x_{si}^2} \quad (7)$$

In summary, in order to calculate the vertical displacement of the linear actuators, one may calculate the s vector using Eq. (4) and then use this result in Eq. (7).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Forward Kinematics Model In the forward kinematics algorithm, the position of the end-effector is calculated based on the given linear actuators positions, measured by absolute encoders.

From Eqs. (4) and (5), we can write:

$$c_i = \|\vec{p} + R \cdot \vec{a}_i - \vec{b}_i - \vec{s}_i + \vec{l}_i\| \tag{8}$$

Which represents a set of six nonlinear equations that composes the forward kinematic model. From Eq. (8) we obtain the position of the end-effector traversing the $\|\vec{m}\|$ distance in the direction perpendicular to the platform ($\hat{v}$ direction). A possible approach for solving these equations is to use a numerical method. In this context, the Newton-Raphson method was chosen for root-finding, as it is widely used in similar problems due to its convergence rate and accuracy performance.

The Newton-Raphson is an iterative algorithm, therefore Eq. (9) defines $f_i(x)$ as the square error between the given actuators position and the calculated position at the kth iteration:

$$f_i(x) = (y_i)^2 - (y_i^{(k)})^2 = 0, \quad i = 1, \ldots, 6 \tag{9}$$

Where the x variable denotes the six input variables vector, representing the end-effector coordinates, and y_i denotes i^{th} output variable, which is the vertical displacement of the ith linear actuator. Let $F(x)$ be the vector of f_i functions. Given an estimate $x^{(k)}$ for the platform centroid position at the kth iteration, the linear approximation around $x^{(k)}$ can be calculated as:

$$F(x) = F(x^{(k)}) + (x - x^{(k)})F'(x^{(k)}) \tag{10}$$

Where $F'(x) = (J)_{ij} = \frac{\partial f_i}{\partial x_j}$. Hence, the Jacobian matrix is defined as:

$$J = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_6(\mathbf{x})}{\partial x_6} \end{bmatrix} \tag{11}$$

Then, the input vector at the $k^{th} + 1$ iteration is calculated as:

$$x^{(k+1)} = x^{(k)} + J^{-1}(x^{(k)})F(x^{(k)}) \tag{12}$$

The algorithm proceeds until the convergence criterion $\|F(x^{(k)})\| < \epsilon$ is met or until the maximum number of 10 iterations is reached.

Kinematics Implementation

The Delta Tau integrated development environment (IDE) supports two different ways to implement the control system kinematics. It can be done using the kinematics scripts (plc files), or implementing in C language, which is indicated when the application requires heavy mathematical calculations, due to the fact that compiled languages have better runtime performance when compared to scripted ones. In this sense, our choice was to implement the C source coded kinematics.

There are some obstacles when implementing the C kinematics in Delta Tau system that had to be overcame. The Delta Tau IDE doesn't support the usage of additional C libraries, only the ones included in the software development kit (SDK), which made it necessary to adapt and use the proprietary functions included in SDK. In addition to that, the C kinematics logics have to be implemented in a specific function already defined in the SDK, the *CfromScript* function, which is called inside the script files.

On the other hand, the SDK includes functions that were useful aiming a motion control better perfomance. The *SegMoveTime* function splits the movement trajectory into time intervals of configured milliseconds. Before each segment execution, the kinematics are re-calculated. This improves the movements accuracy during the transient state by reducing the parasitic movements in other DoFs, although it costs computation resources. If the interval is set too low, there is a risk of starvation, where the controller may not be able to do all of its move calculations in the time allotted, stopping the motion program with a run-time error. As our application targets better positioning precision, we decided to use 1 ms inteval configuration, which has shown to be enough for motion calculations during our tests.

VALIDATION TESTS

The present section outlines the implemented motion control system validation tests, which includes its comparison with Bestec's P494 Motion Control System benchmark. Furthermore, it describes a simple example of the desired arbitrary trajectory functionality implementation.

Performance Tests

Aiming the standardization of control systems for Sirius beamlines' PKMs, such as granite bases, hexapods and tripods, a motion control system was developed in in-House. For the validation of this system, comparative tests were performed against the Bestec P494 motion control system.

These tests consisted of making controlled movements in one hexapod DoF, setting the same motion range and speed in both control systems, while logging all DoF's positions to check the amplitude of parasitic movements. The movements were performed using one control platform at a time. The unit positions were logged, then acquired and analyzed afterwards. Due to the fact that P494 controller sample rate for the hexapod position is limited to 10 samples/s, we configured the Power Brick LV to acquire at the same sample rate.

For compilation purposes, this paper only presents two cases of the conducted comparative performance tests, one representing a translation DoF command and another illustrating a rotation DoF command.

Figure 4 depicts the comparison between the control systems performances when commanding the unit to move from

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

−1 mm to 1 mm in *Z*-axis. The *Z*-axis curve shows that the control systems reached stability at the target position with a difference of approximately 0.4 s. Although P494 control reaches the destination before, it presents significant position deviations at the transient state for all DoFs, while the Power Brick LV shows smaller control errors. The Fig. 4 curves also shows that cross-talk movement ranges were smaller for the Power Brick LV system. In *X*, *Y*, R_x and R_z axes, the noticed differences represent one order of magnitude.

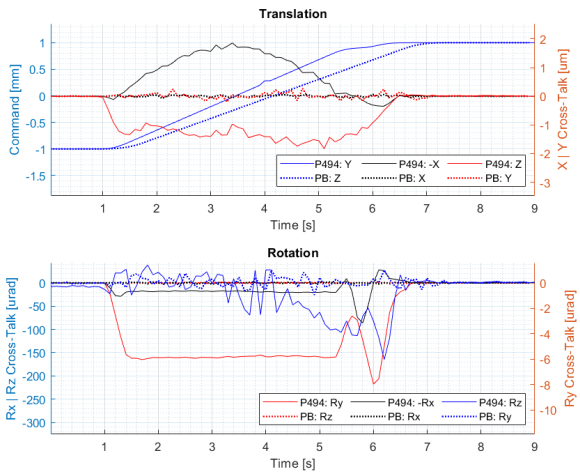


Figure 4: *Z*-axis translation trajectory and parasitic motion comparison between Delta Tau Power Brick and Bestec P494 control systems.

The command of R_z-axis moving from −3 mrad to 3 mrad was the chosen rotation DoF, which is shown in Fig. 5. The R_z-axis curve exposes a similar behavior to the previous test. The P494 system reaches the target approximately 0.3 s before, although it presents position deviations, while the Power Brick course the trajectory without position hiccups. The remaining curves illustrate that parasitic movement ranges are smaller using the Power Brick LV system for all axes. In X and Y cases, the differences represent one order of magnitude. The rotation plot also shows that P464 may present stationary error before the movement, which is noted in $P494 : -Rx$ curve.

Arbitrary Trajectory Test

Many beamline applications are not covered by point-to-point traditional s-curve trajectories. For instance, if coordinated motion is required and the hexapod should follow an external motor towards a polynomial relation, the motion profile will demand velocities' profiles other than smooth trapezoids. Or if a crystal should be rotated to vary the beam incidence angle and to perform a motion profile whose energy derivative is trapezoidal, an arbitrary trajectory is needed. If all virtual axes of an hexapod can be controlled with low cross-talk and also to perform arbitrary trajectories, then a powerful tool for fly-scans was achieved.

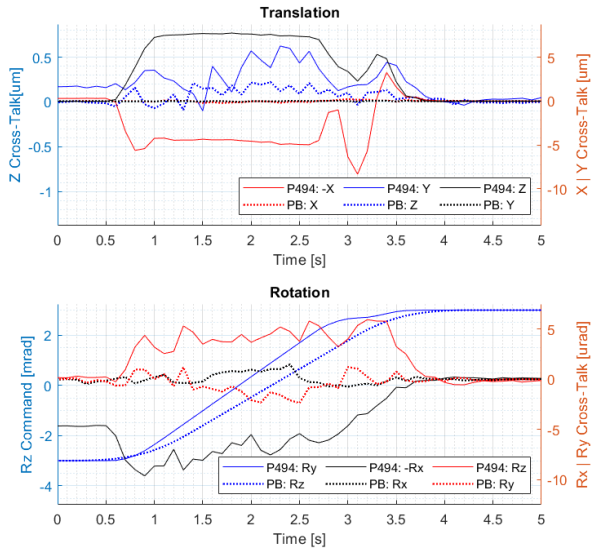


Figure 5: *Z*-axis rotation trajectory and parasitic motion comparison between Delta Tau Power Brick and Bestec P494 control systems.

In this context, the arbitrary trajectory functionality consists of performing system movements based on a previously defined sequence of DoFs positions and movements parameters, such as speed and step triggers. This may enhance the system runtime time performance and determinism, as a portion of the necessary calculations is done before the system motion execution.

When using this functionality, the trajectory parameters may be inputted into the system using different resources, e.g. they may be saved into the device memory and read during application runtime by a script, processed by a motion application. Another option is to use the EPICS interface through waveforms Process Variable (PV) to input the trajectory parameters. This paper presents a simple validation example of the second method, using a customized version of Power Brick LV and EPICS source codes implemented by Diamond Light Source (DLS) [10], which inputs the trajectory positions, time spent in each position and the speed curve mode in each step move.

For this test, the trajectory parameters were previously generated using a python script. Figure 6 illustrates the planned trajectory, which commands the hexapod to move in the 3 translations DoF.

Aiming a more realistic application, the generated trajectory also variates the speed and acceleration during the movement, which may be useful during beamline fly-scans. In this sense, a sinusoidal-exponential signal was chosen as arbitrary acceleration profile, in order to create a constantly varying velocity profile. Each trajectory edge is composed by two axes moves, one is a linear 0.001 mm increment or decrement in the range from 0 to 0.5 mm, and another is described by Eq. (13).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

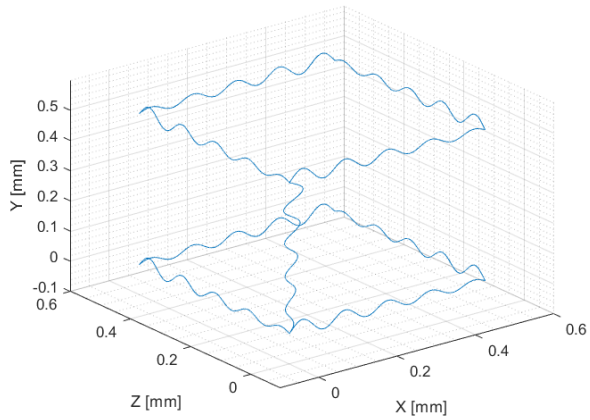


Figure 6: Generated arbitrary trajectory

$$pos_i = \frac{e^{(2*i)} * sin(i) + 2 * i}{100} \quad (13)$$

Figure 7 illustrates the generated position, speed and acceleration data considering the non-linear movement component of a single trajectory edge movement.

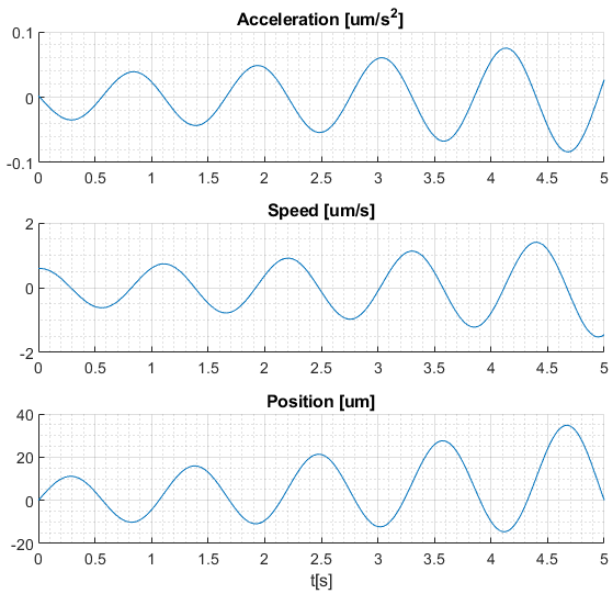


Figure 7: Generated position, speed and acceleration

Based on the generated trajectory, the DLS algorithm was used to run the application and move the hexapod motion unit. Figure 8 depicts the motion system logged position in each translation DoF during the movement, while Fig. 9 shows that the position errors of each commanded DoF during the transient state ranges from −3 to 3 µm.

The analysis of the logged position data in comparison to the planned trajectory, in addition to the low range of the computed errors, shows that the motion system follows the planned motion, which validates the arbitrary trajectory functionality implementation.

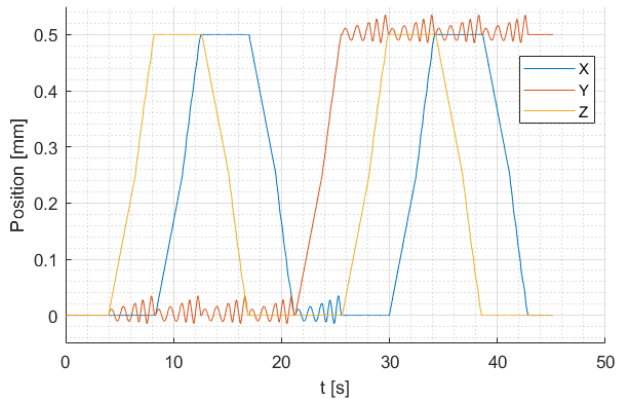


Figure 8: Time plot of arbitrary trajectory logged positions.

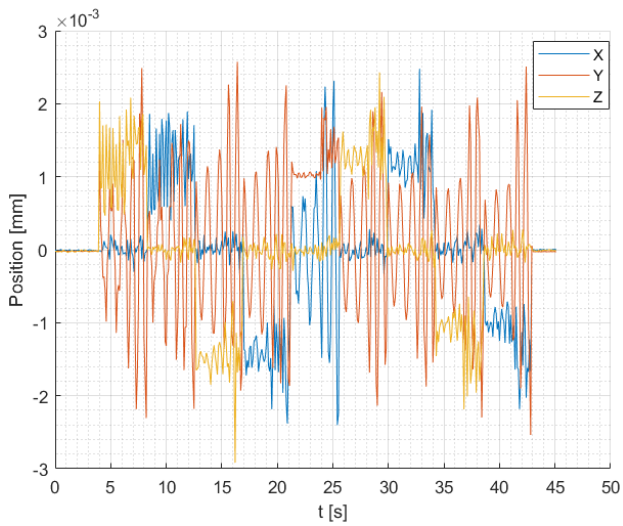


Figure 9: Time plot of transient state position errors.

CONCLUSION

A motion control system for the Bestec P468 Mirror Unit hexapod was implemented using Delta Tau Power Brick LV, a standard motion controller used in Sirius. Although the Bestec P494 system can reach its specifications at steady state and perform step-scans, it is neither suitable for fly-scans nor distributed coordinated motion, as relevant cross-talks are present during commanded movements. The performance tests demonstrate that the in-house designed system's motion control accurary at transient state overcomes the manufacturer one, specially when comparing the parasitic movements during a single DoF command, which is fundamental for fly-scans, where the detectors acquire images during transient motions. In addition, the arbitrary trajectory functionality was validated using the new control system,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

which represents an additional step towards beamline coordinated motion.

Once the Delta Tau Power Brick LV is the standard controller for many Sirius' beamlines, the knowledge acquired within the present work can be applied to other systems, *e.g.* implementation of system kinematics in C language for granite bases [11] and the arbitrary trajectory and coordinated motion functionalities for monochromators [12].

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, Innovation and Communication and the contribution of the LNLS team, and the Omron DeltaTau team for technical support, providing hexapod code examples. Besides, Diamond Light Source for sharing development knowledge through their open-source repositories.

REFERENCES

[1] L. Liu, M. B. Alves, F. H. de Sá, A. C. S. Oliveira, and X. R. Resende, "Sirius commissioning results and operation status," in *Proceedings of the 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, Brazil: JACoW Publishing, May 2021.

[2] L. Liu, X. Resende, and F. De Sá, "A new optics for sirius," in *Proceedings of the 7th Int. Particle Accelerator Conf. (IPAC'16)*, doi:10.18429/JACoW-IPAC2016-THPMR013, Busan, Korea: JACoW, Jun. 2016, pp. 3413–3416, ISBN: 978-3-95450-147-2. DOI: doi:10.18429/JACoW-IPAC2016-THPMR013. http://jacow.org/ipac2016/papers/thpmr013.pdf

[3] *Bestec p468 mirror unit esm nsls-ii*, Aug. 12, 2021. https://www.bestec-berlin.de/2019/optics/product-category/mirror-unit/3983/p468-mirror-unit-ipe-lnls/

[4] J. .-. Merlet, *Parallel Robots*. Springer-Verlag GmbH, Jul. 2006, ISBN: 9781402041334. https://www.ebook.de/de/product/11430591/j_p_merlet_parallel_robots.html

[5] *Power brick lv ims*, Aug. 12, 2021. https://www.faradaymotioncontrols.co.uk/page/power-brick-lv-ims/

[6] J. J. Craig, *Introduction to robotics: mechanics and control*, 3rd. Upper Saddle River: Pearson Education International, 2005.

[7] T. Kluyver *et al.*, "Jupyter notebooks - a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Scmidt, Eds., IOS Press, 2016, pp. 87–90. https://eprints.soton.ac.uk/403913/

[8] M. Beg *et al.*, "Using jupyter for reproducible scientific workflows," *Computing in Science & Engineering*, vol. 23, no. 2, pp. 36–46, Mar. 2021. DOI: 10.1109/mcse.2021.3052101.

[9] J. Ehiwario, "Comparative study of bisection, newton-raphson and secant methods of root finding problems," *IOSR Journal of Engineering*, vol. 4, pp. 01–07, Apr. 2014. DOI: 10.9790/3021-04410107.

[10] Diamond Light Source, *Arbitrary trajectory github repository*, Sep. 14, 2021. https://github.com/dls-controls/PMAC-Trajectory-Scans

[11] G. N. Kontogiorgos, A. Y. Horita, L. M. Santos, M. A. L. Moraes, and L. F. Segalla, "The mirror systems benches kinematics development for sirius/lnls," presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Shanghai, China, Oct. 2021, paper TUPV001.

[12] L. M. dos Santos *et al.*, "The control system of the four-bounce crystal monochromators for sirius/lnls beamlines," presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Shanghai, China, Oct. 2021, paper TUPV003.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR04

GENERIC DATA ACQUISITION CONTROL SYSTEM STACK ON THE MTCA PLATFORM

J. Krasna, J. Varlec, U. Legat, Cosylab, Ljubljana, Slovenia

Abstract

Cosylab is the world leading integrator of control systems for big physics facilities. We frequently integrate high speed data acquisition devices on the MicroTCA platform for our customers. To simplify this process, we have developed a generic control system stack that allows us to support a large set of MicroTCA hardware boards with minimal firmware and software modifications. Our firmware supports generic data acquisition up to 32-bit sample width and also generic data generation. The firmware modules are implemented in a way so that support for MRF timing modules can be added and allow the board to act as a MRF timing receiver. On the software side we implemented the control software stack in NDS which means that we offer support for EPICS and TANGO control system out of the box.

SYSTEM DESIGN

Cosylab had worked on multiple high-performance data acquisition (DAQ) solutions throughout the years and some of those DAQ solutions, like applications for beam Current monitors, beam profile monitors, LLRF, etc. had overlapping core functionality. Because of this an challenge arose to determine if a generic control system stack that would cover this fundamental, overlapping functionality could be implemented.

The main objective was to design a system that would improve code reusability and reduce time to support new hardware without sacrificing the ability to extend functionality for custom use-cases. Additionally, the requirement was to cover as many control system frameworks as possible and make the system agnostic to specific frameworks. One of the desired outcomes was to reduce expertise needed to support existing and custom use-cases and create a central, standardized, mature codebase that all engineers could learn from.

Hardware

Looking at the generic form-factors currently available on the market that support DAQ use cases, especially for high-end applications, we can safely say that MicroTCA [1] has the most vendors supporting it and it is growing in popularity on the user-side as well. It is in use at different labs around the world for the already mentioned use cases (BCMs, BPMs, LLRF) and is commonly selected as the go-to platform for DAQ applications. The reasoning is high performance, customizability options through rear transition module expansions as well as advanced features, like IPMI [2] support and others. By working with multiple MicroTCA vendors, for example Cosylab and Teledyne developed DAQ drivers that are used at ITER [3], we have gained a lot of experience and insight into the platform. When selecting a hardware platform for our own product, a Dose delivery system used in medical particle accelerators for cancer treatment, the decision to go towards MicroTCA was a straightforward one.

Figure 1: Vadatech AMC523 module.

After careful consideration which hardware was most appropriate for our general DAQ system or gDAQ for short, we decided to use AMC523 [4] from Vadatech as well as the MRT523 rear transition module [5] and MZ523B mezzanine [6], see Fig. 1. The boards support 12 analog input channels with variable gain, 2 analog output channels and can sample at 125 megasamples per second.

Software

The application driver was implemented using NDS3 or Nominal Device Support [7] which is a public, open-source library that was developed at Cosylab to simplify integration of DAQ hardware for a variety of control system frameworks but is targeted mainly for EPICS [8] and TANGO [9]. The first iteration was built with the focus on EPICS, see Fig. 2. All GUI components were developed using Control System Studio Phoebus [10], see Fig. 3.

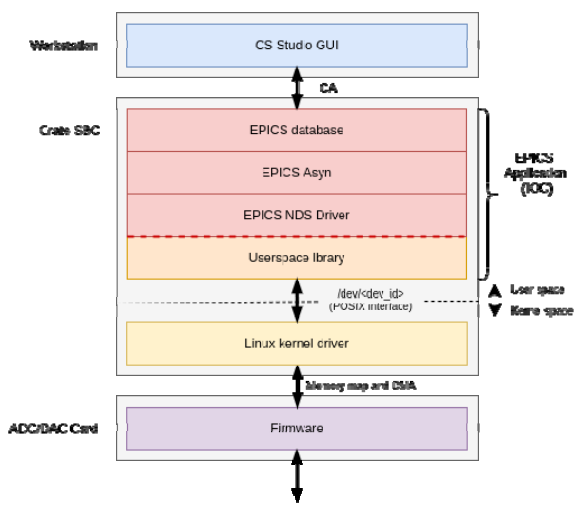


Figure 2: System architecture of the EPICS application.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR04

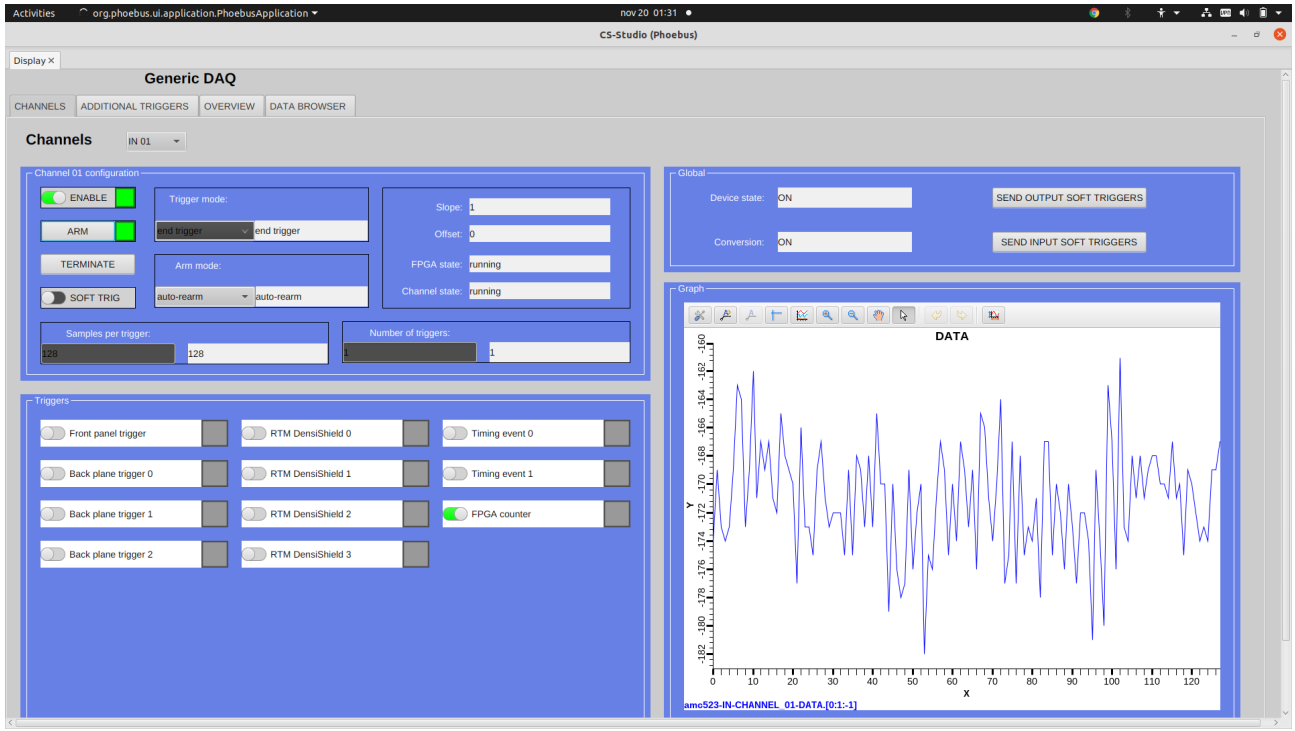


Figure 3: System GUI were implemented using the Control System Studio Phoebus.

In order to write a system that is hardware-agnostic and generalized we moved most hardware-specific parameters to a JSON [11] configuration file which is used by all components in the stack to adapt to a specific hardware board. The firmware is also configured by pushing the configuration data that is also stored in the JSON file through the userspace library.

The kernel driver was designed to include as little functionality as possible to reduce the amount of code running in the kernel and thus decreasing the likelihood of causing a kernel crash due to bugs or errors. The userspace library has a generalized but still rich and user-friendly C++ API which allows direct integrations, avoiding NDS altogether. The userspace library was written with extensibility in mind by exposing the firmware register set which makes adding support for new firmware logic simple and straightforward.

Firmware

Nominal Device Support provides a general abstraction of the device and its signals and while it supports both EPICS and TANGO out of the box it can be extended to support other frameworks as long as the tree-like description of device and its channels and the state machine logic are versatile enough for the new interface. The main feature of NDS that made this generic stack possible is that NDS generates the EPICS database and TANGO device servers programmatically which enabled using the JSON configuration as input during generation.

Firmware was designed and implemented to support a wide variety of DAQ features that can be found on firmware modules from other vendors:

- Configurable data width up to 32-bit
- Configurable up to 32 data acquisition and 32 signal generation channels
- Configurable buffer allocation for each channel separately
- Separate enable/disable for each channel
- Multiple trigger sources including software
- Arm and automatic re-arm
- Trigger mode: Front trigger, end trigger (prebuffering)
- Multiple data generation sources: from RAM, internal data stream (NCO), digitally processed ADC data
- Data streaming options: on trigger, periodic

We are also confident that firmware can be ported to other hardware platforms with minimal changes as long as a similar FPGA chip is available as the one used on the AMC523 (Kintex-7) although no such ports are planned as of this moment.

Timing Support

In order to support timing functionality an FPGA core was developed that implements Micro Reseach Finland (MRF) [12] receiver functionality. This core can be included on DAQ boards that provide an SFP or other compatible connector and in this way allow the timing to be connected directly to the board. The more typical use-case of having a separate MRF receiver and triggering DAQ over the backplane or through a front-panel connector is also supported out of the box.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONCLUSION

The gDAQ system was developed, tested and works as expected. We were able to cover all the generic functionality and therefore have a starting baseline for future projects. The system is currently in a prototype phase and was not yet deployed to a production environment.

REFERENCES

[1] MicroTCA standard overview, https://www.picmg.org/openstandards/microtca/

[2] IPMI Specification, V2.0, Rev. 1.1: Document, https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html

[3] ITER Project, https://www.iter.org/

[4] Vadatech AMC523 DAC module, https://www.vadatech.com/product.php?product=384

[5] Vadatech MRT523 Rear Transition Module, https://www.vadatech.com/product.php?product=487

[6] Vadatech MZ523B 12 channel ADC module, https://www.vadatech.com/product.php?product=485

[7] NDS – Nominal Device Support v3, https://github.com/Cosylab/nds3

[8] EPICS – Experimental Physics and Industrial Controls System, https://epics-controls.org

[9] TANGO open-source SCADA and DCS, https://www.tango-controls.org/

[10] Control System Studio (Phoebus), https://controlssoftware.sns.ornl.gov/css_phoebus/

[11] JSON – JavaScript Object Notation, https://www.json.org/json-en.html

[12] Micro Research Finland Oy, http://mrf.fi

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV001

STATUS OF THE SARAF-PHASE2 CONTROL SYSTEM

F. Gougnaud†, F. Gohier†, P. Bargueden, D. Darde, G. Desmarchelier, G. Ferrand, A. Gaget, P. Guiho, T. Joannem, A. Lotode, Y. Mariette, S. Monnereau, V. Nadot, V. Silva, N. Solenne CEA Saclay IRFU, Gif sur Yvette, France
E. Reinfeld, I. Shmuely, H. Isakov, A. Perry, Y. Solomon, N. Tamim, T. Zchut, SNRC, Yavne, Israel

Abstract

SNRC and CEA collaborate to the upgrade of the SARAF accelerator [1] to 5 mA CW 40 Mev deuteron and proton beams and also closely to the control system. CEA is in charge of the Control System (including cabinets) design and implementation for the Injector (upgrade), MEBT and Super Conducting Linac made up of 4 cryomodules hosting HWR cavities and solenoid packages.

This paper gives a detailed presentation of the control system architecture from hardware and EPICS software points of view. The hardware standardization relies on MTCA.4 that is used for LLRF, BPM, BLM and FC controls and on Siemens PLC 1500 series for vacuum, cryogenics and interlock.

CEA IRFU EPICS Environment (IEE) platform is used for the whole accelerator. IEE is based on virtual machines and our MTCA.4 solutions and enables us to have homogenous EPICS modules. It also provides a development and production workflow. SNRC has integrated IEE into a new IT network based on advanced technology.

The commissioning is planned to start late summer 2021.

CONTEXT OF THE PANDEMIC

Because of the pandemic, the CEA team hadn't been able to go to the SNRC Lab since March 2020. The first impact was for the Injector. The CEA is in charge of the EPICS update of the Injector Control System. Four cabinets were delivered to SNRC in February 2020 as planned but the CEA could no longer go there and install software. Therefore, SNRC installed the four new cabinets and integrated the Injector control software by itself. At that time a remote connection between the 2 labs was not allowed. The CEA support was only by videoconferences and emails. Finally, there was a first beam on the Source on September 2nd 2021.

GENERAL ARCHITECTURE

In summer 2018, our partner SNRC accepted CEA's recommendation to migrate to the CEA MTCA.4 platform for the SARAF control system. This platform was presented at ICALEPCS19 [2] and is based on the very compact NATIVE-R2 crate with this common core on each crate: the NAT-MCH-PHYS80 board offers an 80-port PCIe Gen3 switch and can be combined with the Rear Transition Module CPU NAT-MCH-RTM-COMex-E3.

† francoise.gougnaud@cea.fr , Francis.gohier@cea.fr

For semi-fast and fast acquisition, a set of IOxOS boards was also added to this common platform. See Figure 1. The intelligent FMC carrier IFC1410 (AMC form factor featuring an NXP QorIQ T series Power PC processor and one Xilinx Kintex UltraScale FPGA accessible by the user) is used with ADC-3117 and ADC-3111 FMC boards. The fast acquisition board ADC-3111 includes 8 channels with 16-bit/250Msps ADCs. The semi-fast acquisition board ADC-3117, whose sampling frequency is comprised between 1 and 5 Msps, has 20 channels and 2 channels DAC.

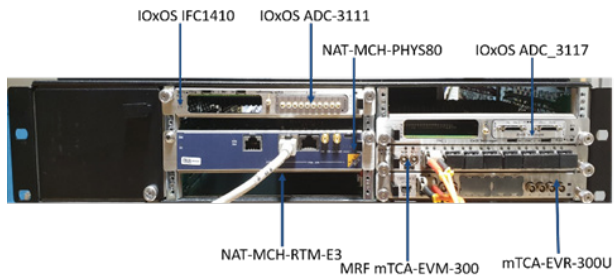


Figure 1: MTCA.4 NATIVE-R2 standardized platform.

The CEA team updated and standardized the IRFU EPICS Environment with MTCA.4 solutions based on IOxOS, MRFand NAT boards.

For the PLC domain essentially used for vacuum, cryogenics, tuning motorizations, current leads and interlocks, Siemens PLC 1500 series (CPU 1516) was selected with TIA Portal by CEA team.

Kontron Industrial PCs are also used in order to run EPICS IOCs for the communication with PLCs. This communication is based on Modbus/Tcp and S7plc.

FARADAY CUPS, ACCTS AND NBLMS

On LEBT and MEBT, this common platform including IOxOS boards is used for Faraday Cups and ACCTs intensity measurement with the IOxOS ADC-3117 board.

An acquisition based on IOxOS ADC-3111 is used for the Neutron sensitive Beam Loss Monitors (gaseous detectors) designed by CEA for SARAF. This CEA design is already used for ESS and gives entire satisfaction.

TIMING SYSTEM

The Timing System distributes trigger signals including the information for each RF pulse and manages timestamping mechanism that allows to date all the actions and events precisely. This timestamping is internally incremented and is periodically resynchronized with the Meridian II GPS by

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the PPS (Pulse Per Second) clock to avoid drifts. The Timing System also distributes the 176 MHz reference frequency coming from the Master Oscillator.

MRF solution was decided early in the project. The timing system modules used are the MRF mTCA-EVM-300 used as Event Master and the MRF mTCA-EVR-300U as Event Receiver. In addition, mTCA-EVM-300 modules are used as fan-outs and concentrators. There is one EVR in each MTCA crate and one fan-out/concentrator almost in each one.

MTCA.4 backplane lines and Timing System signals have been standardized for all the MTCA EVRs, it means LLRF, BPMs, nBLMs and ACCT/Faraday Cup acquisitions as indicated in table 1.

The SARAF timing system is detailed in [3].

Table 1: MTCA.4 Backplane Lines and Timing Signals

Backplane standardization		
Crate bus line	Trigger code	Trigger nick-name
17_1	TS1	Trigger user buffer (TUB)
17_2	TS2	Beam presence or Acquisition (RoI)
18_1	TS3_m	Signals dedicated to the protection (m = instance of LLRF, BPM …)
18_2	TS3_m	
19_1	TS3_m	
19_2	TS3_m	
20_1	TS4	RF gate
20_2	TS6	General post mortem (GPM)

LLRF SYSTEMS

LLRFs have been outsourced to Seven Solutions [4] with the duty to use the common core (NATIVE-R2, COMex-E3, PHYS80 and MRF EVR) in the purpose of homogeneity of the control architecture. Seven Solutions has designed an AMC board running 2 LLRFs and the standardized NATIVE-R2 can only include 2 LLRF AMC boards. Seven Solutions has designed, developed, manufactured and tested the system based on CEA technical specifications. The final version of this digital LLRF will be installed in the SARAF accelerator in Israel at the end of 2021.

RFQ and MEBT sections need four LLRFs, one for the RFQ and three for the 3 rebuncher cavities. In the Super Conducting Linac, there are 4 cryomodules including 6 and 7 cavities respectively for CryoModule 1 and CryoModules 2, 3 and 4. Therefore, there are 2 NATIVE-R2 crates for each cryomodule and one for the set RFQ and MEBT. Globally 31 LLRFs are needed for the SARAF Linac.

Currently, the LLRF tests give satisfaction and the MEBT RF cabinet (Figure 2) is ready for the shipment to SNRC.

Figure 2: RF cabinet with MTCA LLRF crate.

BEAM POSITION MONITORS

The beam position measurement function is spread over the linac with 24 BPMs and it is performed by analysing signals of the 4 buttons of each BPM, a total of 96 signals. The layout strategy is to group the BPMs operating at room temperature and those operating at temperatures close to 4 K degrees, respectively 4 in the MEBT and 20 in the 4 cryomodules of the superconducting linac (6 for CM1 and CM2, 4 for CM3 and CM4).

For BPMs, some requirements are very close to LLRF issues and finally the BPM control has also been outsourced to Seven Solutions.

BPM systems are included in MTCA NATIVE-R2 crate. Each one can include up to 3 BPM boards, each board handling signal analysis of 2 BPMs. Overall, the linac BPM function is ensured by 5 MTCA crates: one BPM crate for the MEBT and one crate for each cryomodule. The BPM control development is in progress.

HARPS

There are 5 Harps in the MEBT and SCL.The Harps and their electronics are outsourced to the Proactive company. They don't have an MTCA solution but only a VME solution. The VME will house 16 digitizer cards that will communicate with an EPICS standardized Kontron IPC through a USB hub. This IPC will include a PCIe-EVR 300DC for integration in the MRF Timing System network. This EVR300 is connected through a microSPCI card to an IFB300 box that delivers up to 16 TTL signals. See Figure 3.

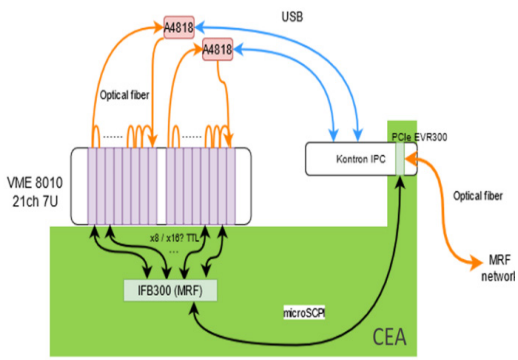


Figure 3: Harps control based on VME, Kontron IPC and MRF.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV001

MEBT QUADRUPOLES, STEERERS AND SCL SUPERCONDUCTING MAGNETS

Different CAENels power supplies were chosen for all these magnets. There are 8 quadrupoles and steerers in the MEBT and 20 superconducting solenoid in the SCL. For the CAENels power supplies, the EPICS control is based on streamdevice and TCP/IP communication. Interlocks are managed by PLC. This control was installed at SNRC in 2020 for the MEBT.

CONTROL OF THE 4 CRYOMODULES

The SARAF Super Conducting Linac offers four CryoModules (CM) including 6 cavities for CM1 and 7 cavities for CM2, CM3 and CM4 and 20 superconducting solenoids (6 solenoids for CM1 and CM2 and 4 for CM3 and CM4). Each cryomodule can be divided into four control type applications: cryogenics, vacuum, solenoid current lead and LLRF cold tuning system.

One PLC CPU manages every sensor, actuator and automatic processes through the remote input-output cards and also the communication protocols (TCP-IP or Modbus) and fieldbuses like Profinet and Profibus DP.

The field network Profinet is essentially used to control cryogenic temperatures, remote input-output channels, motorizations based on Phymotions [5] and Festo devices for pneumatic valves. Profibus DP is mainly used for vacuum controls (turbopump controllers and total pressure gauges TPG300 controllers).

This control architecture based on only one PLC CPU per cryomodule is duplicated for each one of the four cryomodules for homogeneity purpose but also to simplify the communication with the SOREQ helium liquefier.

Each cryomodule PLC will have to deal with data of the SNRC local liquefier and this will be done via one PLC interface that will be the unique communication partner of the Air Liquide liquefier PLC.

For each cryomodule, a Kontron Industrial PC is running an EPICS IOC that enables the communication between the PLC CPU with any other EPICS IOC of the architecture (Figure 4).

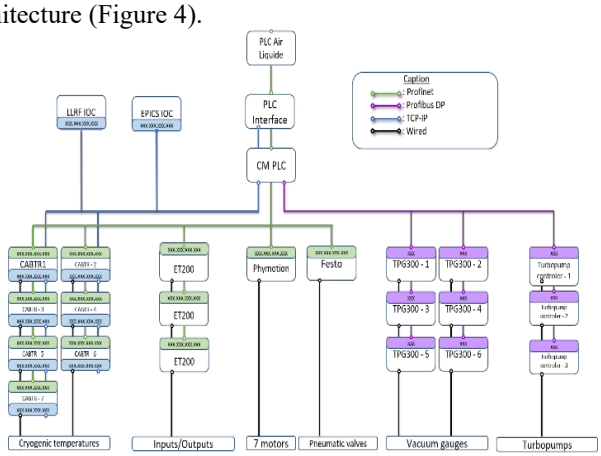


Figure 4: Architecture template for 1 of the 4 CryoModules.

SARAF EPICS ENVIRONMENT

Irfu EPICS Environment

The SARAF project uses the Irfu EPICS Environment (IEE) to deploy a homogenous and versioned EPICS environment. As detailed in a previous publication [6] the IEE relies mainly on the following aspects: a centralized EPICS installation is available from an NFS server; CPU devices use stateless network boot to load the same generic root file system; virtual machines (VM) are used for robustness and eased sharing; Ansible scripts under continuous integration enable to provision and update the IEE server and client machines. See Figure 5.

IEE Ansible Scripts

IEE Ansible scripts were delivered in December 2020 and used successfully on premises at SNRC to provision the current production IEE server. The main difficulty has been that the accelerator network is not connected to the Internet, this was solved by provisioning the IEE server on another network before moving it to the production network through a cybersecurity check. This could be improved in the future by deploying a Yum repository server on the premises. Similarly, this lack of internet made the delivery of git packages more complicated, but also because the CEA Gitlab instance is not open to non-CEA users, this has been solved by setting up a continuous delivery pipeline to duplicate and keep updated the appropriate repositories on a shared Bitbucket-cloud instance. This bitbucket instance is then copied locally at SNRC onto the secured network.

Running Ansible

Running Ansible scripts is not the only way used to distribute the IEE environment. To sum up, in practice only system administrators use the scripts: either to provision the production environment within virtual containers (CT) in Proxmox or Nutanix clusters, and also to provision “ready-to-use” VM for use with VirtualBox. These VMs are for instance shared with subcontractors to ensure the compatibility of their developments to the IEE, or by developers that need to have a working environment disconnected from the IEE server. VMs are also provided for acceptance tests, this way new hardware and software can be validated before being fully integrated into the production network. All these VMs provide the feature of an IEE server (local EPICS environment and if needed boot server) as well as an IEE DevEnv machine (users, CSS, VSCode [7]…). They are produced by running the same Ansible playbook that can configure either IEE servers or IEE DevEnv.

Unlike CTs running on a cluster, which are provisioned using an orchestration machine to run the Ansible playbook, on VM this playbook is executed locally (after installing Ansible on the VM later on). Therefore, the scripts are still available to update the VM easily. This is particularly useful when the VM is also used as a boot server.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

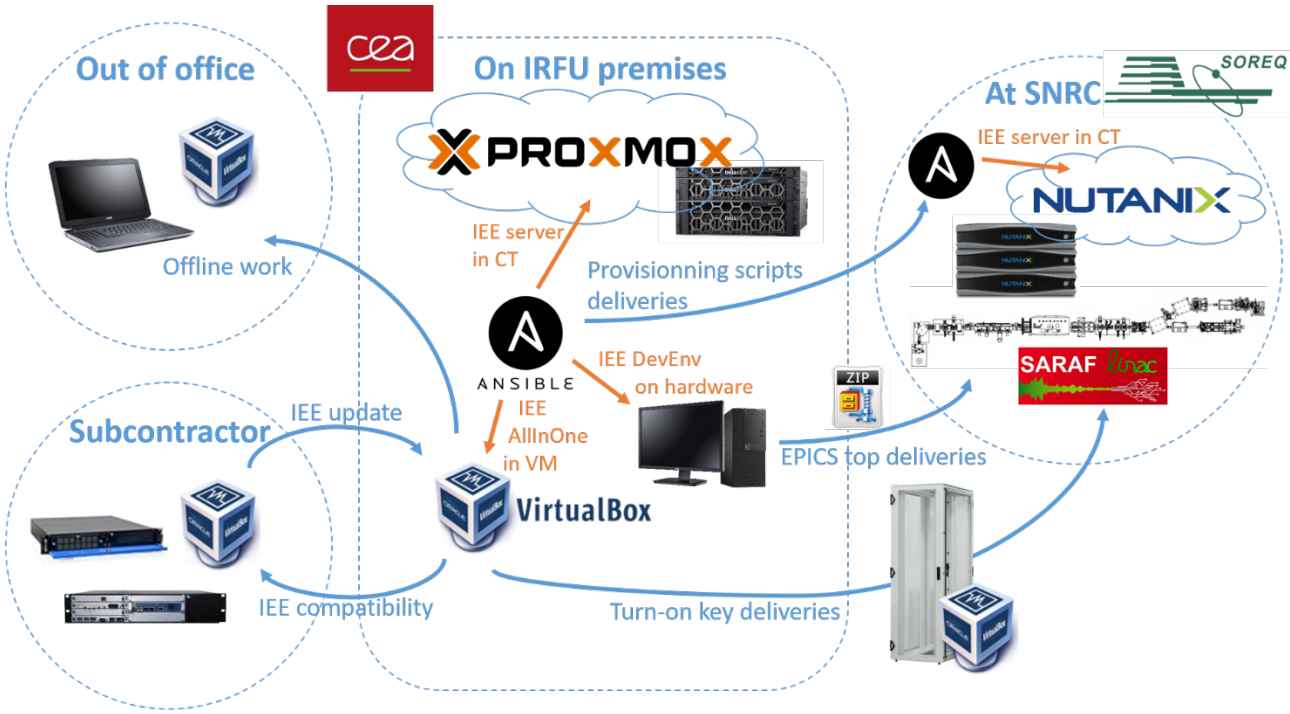


Figure 5: CEA and SARAF EPICS environment.

Improvements

However, the ability to use the Ansible scripts to update, rather than configure a machine from scratch, has only been improved recently. This combined with the need to provision the machine on a separate network before moving it to the secured network, has made it such that producing a new IEE Server, was not fast enough to follow the delivery rate of developers. Therefore, bug fixes and major feature request of EPICS development are delivered by sending a top composed of git submodules.

Prospects

Another drawback of Ansible scripts is that integrating new functionalities can be very time consuming, even if this change is anecdotal when performed manually. Automating is often costing when it must be done properly (researched, implemented, tested and CI maintenance). For this reason, outsourced developments have a pretty loose constraint on what it means to be compatible with the IEE environment. In the end it mainly involves using stateless network boot, and using a predefined NFS mount to export data. The EPICS version is fixed, and a naming must be followed. But cross-compilation tools and code source are not required to be integrated to the IEE Server unlike for CEA developments.

CONCLUSION

The SARAF Linac control includes many new technologies as described herein that are very motivating for SNRC and CEA control teams. Furthermore, the relationship is very good between the two teams. The Injector and MEBT are already located at SNRC and the control tests are in finalization or in progress depending on the devices.

During the coming months, the CryoModule 1 will be tested on the CryoModule Test Stand at Saclay. Then, the four Super Conducting Linac cryomodules including cavities and solenoids will be shipped from CEA Saclay to SNRC during 2022.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV001

REFERENCES

[1] N. Pichoff et al., "The SARAF-LINAC Project 2019 Status", in *Proc. 10th Int. Particle Accelerator Conf. (IPAC'19),* Melbourne, Australia, May 2019, pp. 4352-4355. doi:10.18429/JACoW-IPAC2019-THPTS116

[2] F. Gougnaud et al., "Evolution Based on MicroTCA and MRF Timing System", presented at the *17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19),* New York, NY, USA, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-MOPHA052

[3] A. Gaget, "MRF Timing System Design At SARAF", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper THPV022, this conference.

[4] J. Fernández, P. Gil, J. G. Ramirez, G. Ferrand, F. Gohier, and N. Pichoff, "Status of the uTCA Digital LLRF design for SARAF Phase II", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper WEPV031, this conference.

[5] T. J. Joannem et al., "Motorized Regulation Systems for the SARAF Project", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper TUPV007, this conference.

[6] J. F. Denis, F. Gohier, A. Gaget, F. Gougnaud, T. J. Joannem, and Y. Lussignol, "IRFU EPICS Environment", presented at *the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19),* New York, NY, USA, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-WEPHA040

[7] V. Nadot, A. Gaget, F. Gohier, P. Lotrus, and S. Tzvetkov, "vscode-epics, a VSCode Module to Enlight Your Epics Code", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV024, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

CENBG CONTROL SYSTEM AND SPECIFIC INSTRUMENTATION DEVELOPMENTS FOR SPIRAL2-DESIR SETUPS

L. Daudin†, P. Alfaurt, A. Balana, M. Corne, M. Flayol, A. Husson, B. Lachacinski, CNRS/IN2P3
Université de Bordeaux CENBG, Gradignan, France

Abstract

The DESIR facility will be in few years the SPIRAL2 experimental hall at GANIL dedicated to the study of nuclear structure, astrophysics and weak interaction at low energy. Exotic ions produced by the S3 facility and SPIRAL1 complex will be transferred to high precision experiments in the DESIR building. To guaranty high purity beams to perform high precision measurements on specific nuclei, three main devices are currently being developed at CENBG: a High Resolution Separator (HRS), a General Purpose Ion Buncher (GPIB) and a double Penning Trap named "PIPERADE". The Control System (CS) developments we made at CENBG are already used to commission these devices. We present here beamline equipment CS solutions and the global architecture of this SPIRAL2 EPICS based CS. To answer specific needs, instrumental solutions have been developed like PPG used to optimize bunch timing and also used as traps conductor. Recent development using the cost efficient Redpitaya board with an embedded EPICS server will be described. This device is used to drive an FCup amplifier and is also used for particle counting and time of flight measurements using our FPGA implementation called "RedPiTOF".

THE DESIR FACILITY

Overview

In 2024 DESIR [1, 2] is planned to be the new low energy SPIRAL2 facility dedicated to the study of nuclear structure, astrophysics and weak interaction at GANIL (Caen, France). This experimental building will accept low energy (10-60 keV) RIB beams from both historical SPIRAL1 complex [3], delivering heavy ions beams since 1983 and the new SPIRAL2 linear accelerator [4] via the S3 facility [5] delivering high intensity light ion beams since 2019 (see Fig. 1). In DESIR, specific exotic ions will be separated in mass and transferred to high precision experiments to perform decay spectroscopy, laser spectroscopy and mass spectrometry.

CENBG Developments

In order to provide highly purified beams previously introduced, three main devices have been entirely developed and are currently tested and commissioned at CENBG: a High Resolution Separator (HRS) [6], a RFQ-cooler-buncher called "General Purpose Ion Buncher" (GPIB) [7] and a double Penning Trap named "PIPERADE" [8].

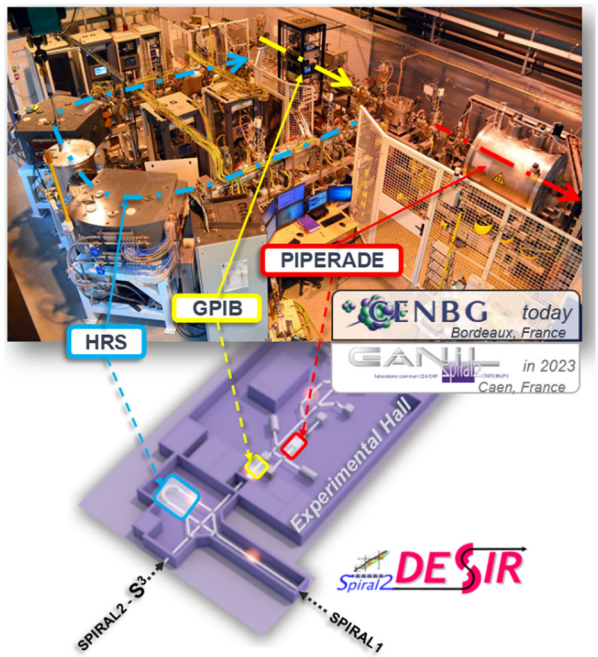


Figure 1: The CENBG setups today at CENBG and the future DE-SIR experimental area, coupling the GANIL building and the new SPIRAL2 facility.

The main concepts of the control system (CS) developed and currently used to test and commission the HRS, the GPIB and PIPERADE is presented in this paper. These CS developments done at CENBG will be extended to the entire DESIR project, meaning to the four DESIR transfer lines (180 meter long):

1. LS: beam transfer from S3 to the DESIR Hall.
2. LT: beam transfer from SPIRAL1 to the DESIR Hall.
3. LHR (High Resolution beamline) including a dedicated RFQ-Cooler SHIRAC (LPC-Caen) and the HRS.
4. LHD (DESIR Hall Beamline): Central delivery "fish-bone" line inside the DESIR Hall.

Milestones

The DESIR beam-lines and the experimental Hall equipped with the first group of experiments are expected in 2023. The HRS separator and PIPERADE traps are already in the commissioning phase at CENBG using the first version of the DESIR control system.

DESIR CONTROL SYSTEM (CS)

SPIRAL2 Collaborative Developments

The DESIR CS and Automation developments for the whole beam-lines and purification devices including the

† laurent.daudin@in2p3.fr / ORCID(s) 0000-0002-3007-4217

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

PIPERADE apparatus are done at CENBG. In order to ensure the compatibility with the already existing SPIRAL2 LINAC accelerator and NFS, S3 experimental areas, these developments are carried out in close collaboration with the GANIL laboratory. It is based on the Experimental Physics and Industrial Control System (EPICS) architecture [9], basic framework for the SPIRAL2 control system [10, 11]. Among the accepted common rules is the naming convention used at SPIRAL2 applying to equipment names as well as CS Server names including EPICS process variables.

EPICS servers (IOC) are developed within the "topSP2" common software platform based on the CentOS Linux system running on PC configured like SP2 CS computers. All of them run the same EPICS base 3.14.9 and share versatile IOCs/EPICS modules dedicated to various equipment like GANIL beam profilers (harp wire monitors) or Hazemeyer power supplies for the HRS electromagnetic dipoles.

Up to now our CS software developments are directly saved and versioned on the GANIL SVN server and a migration to the Gitlab server of the CNRS/IN2P3 institute is planned next year. Software development tools are shared with the GANIL CS group like the Spiral2 version of CSS/BOY (CSS-Dev & CSS-Op) developed under ECLIPSE IDE [11] and used to build most of GUIs. Phoebus [12], the current variant of CSS not depending on Eclipse RCP, is planned to be used in the future to develop DESIR beamlines HMI. Some "Rich" Clients like the "ProfilSP2" Java Application developed at GANIL also operates at Bordeaux during the commissioning phase. Finally, the DESIR beam-lines IOCs for PLC driven equipment communicating with servers using Modbus-TCP protocol will be generated by the GenIOC utility [11].

CS Architecture and Main Options Followed

The DESIR CS architecture is based on EPICS Client-Server system deployed on an Ethernet network using the Channel Access protocol. The management of the vacuum and interlock systems is made via automation as detailed in the following paragraph. This architecture is illustrated in Fig. 2 with the PIPERADE CS example.

For PIPERADE, some equipment are localized on a high voltage platform (~ 30 kV) because the Penning trap itself had to be at this potential to decelerate ions coming in. In this case, unmanaged Ethernet Moxa switches are used with fibber optic port to guaranty the galvanic insulation of equipment without reducing communication.

Most of the EPICS servers are executed as services on a Linux (CentOS) Machine. For DESIR, embedded IOCs have been considered for High Voltage Power Supplies (HV-PS) and Redpitaya boards described later in this paper.

VME crates with embedded EPICS servers are still used in SPIRAL2 for historical reasons to answer high speed data acquisition needs, but also low speed digitizing, digital I/O and analogue signal generation. In 2018, the DESIR project decided not to use VME at DESIR.

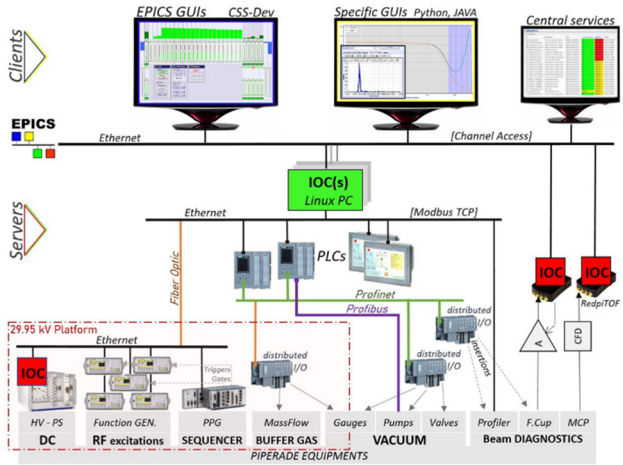


Figure 2: Overview of the PIPERADE CS architecture. Equipment are illustrated on the lower part of this figure. These hardware are controlled by PLCs (Automation part) or directly by EPICS servers (IOC). Ethernet network is the main communication media and Fieldbuses like Profibus and Profinet (Profibus on Ethernet network) are used by PLCs. On the upper part of this figure, Clients are the applications used to control the PIPERADE experiment.

An innovative solution, as detailed in the following automation section of this paper, has been experimented and validated at CENBG to replace VME boards regarding the low speed Faraday Cup (FCup) control and signal digitizing.

Automation Overview

All SPIRAL2 vacuum systems and interlocks are managed using dedicated PLCs (Siemens S7-1500 series) [13]. They are similar to the systems developed by the GANIL CS & Automation group except some improvements hereafter exposed.

As figured in Figure 2, each beamline will have its own Vacuum PLC and Interlock PLC coupled with their own touch panel (local HMI). DESIR PLCs will be localized in a control room far from equipment and will communicate via the Profinet network with distributed Input/Output (I/O) modules (ET200S) deployed in the racks along beamlines. When ET200S are needed on high voltage platforms like the GPIB or PIPERADE, "classical" Ethernet fibber optics are used.

According to the SPIRAL2-Phase1 standard (LINAC, NFS and S3), a "vacuum" PLC unit manages all pumping groups (Pumps, Gauges and Valves) and gas systems (like mass-flow controllers) for a beamline sector. In parallel, Profibus fieldbus allow to collect "live" information from turbo-molecular pumps like the rotation speed or detailed status or fault words.

In addition to the vacuum PLC, a dedicated "interlock" PLC is also affected to each beamline. It interlocks the equipment according to the safety conditions like veto on the high voltage power supplies but also controls all beamline insertion devices like electro-pneumatic translators to put diagnostics IN or OUT the beam axis.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Modbus-TCP servers are implemented in PLCs to manage the communication with dedicated IOCs. The vacuum Modbus table is made of read-only Registers. On EPICS side, only the valves status and pressure values are exchanged and displayed on operator interfaces. Actions on vacuum devices are only authorized using PLCs touch panels, even in remote mode with VNC Clients. WinCC client software will be used at GANIL to develop centralized vacuum interfaces.

New PLC Based Controlled Beamline Equipment

DESIR Interlock PLCs are driving two other kind of equipment: Faraday cups (FCup) and motors.

DESIR beamline FCups are used with "PicoLIN" transimpedance amplifiers [14] developed at GANIL for CW low intensity beam measurement: 8 ranges from 10µA/V down to 1pA/V are selectable. This amplifier is directly controlled in remote mode with the Interlock PLC ET200S Digital I/O and the output signal is sampled and digitized with a 16bit ADC from the same ET200S. This solution used for the HRS tests and commissioning since 2017 is adopted for all DESIR FCups.

The other "new" equipment controlled by PLCs are motors used for the HRS Slits positioning (20 motors), and will be used for six electrostatic 90° beamline deflectors to place or remove them from beam axis. Brushless motors have been selected (Siemens ref 1FK7015) and equipped with a brake to maintain the position when the motor power is switched off (after 1 minute timeout). Classical position encoders with optical disk are not radiation hardened. This is the reason why we preferred these motors with multipole resolvers. Each motor is driven with a Siemens CU-310 one axis control unit and a Siemens PM340 power module. The communication with the Interlock PLC is based on Profinet fieldbus.

SPECIFIC SOLUTIONS FOR DESIR EQUIPMENT

"Heavy" Client's Developments

The EPICS CA protocol makes it possible to monitor and control any beam line equipment through PV calls in many programming languages.

Using EPICS features, a Python program is under development to operate the PIPERADE traps, i.e. scan the different trapping and excitation parameters, define the time sequence and treat on-line measurements. It is inspired by the PyMassScanner program developed by the JYFLTRAP group (Jyväskylä University, Finland) [15] part of our collaboration.

Other Python software tools have been developed at CENBG to answer tests and commissioning needs for the GPIB and PIPERADE. The first example is "Plotpot" which displays the electric field seen by ions in Paul or Penning traps with respect to voltages applied on *the* electrodes. A second example is the automatic beam injection with a multiparameter algorithm based on a simplex optimization method.

"CorrAb" is another tuning application under development at CENBG for the HRS optimization. It will be used to estimate the separator aberrations using a dedicated emittancemeter and then apply the computed voltages to the 48 poles of the HRS multipole to correct as much as possible the separator aberrations. The goal is to reach a resolution of 20000 for isobaric separation using this software.

High Voltage Power Supplies (HV-PS)

Most of the DESIR beam-lines and setups are composed of electrostatic optics: Quadrupoles, Hexapoles, Benders and Steerers.

In order to meet the High Voltage-Power Supplies (HV-PS) requirements (optics and diagnostics), ISEG multichannel crates provide all DC voltages (see Fig. 3).

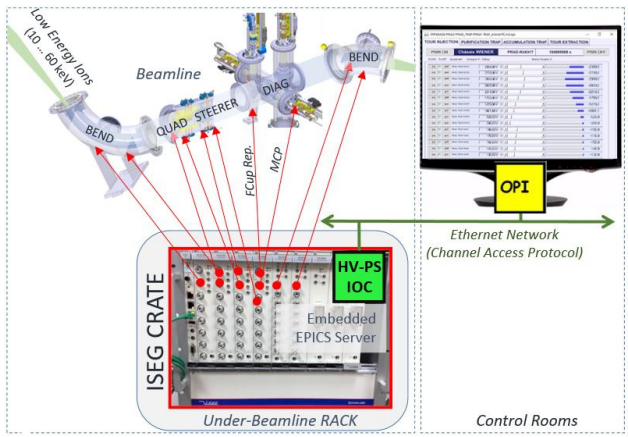


Figure 3: Hundreds of high voltage power supplies (HV-PS) will be deployed under DESIR beamlines and setups. Each multichannel ISEG crates will concentrate a large number of HV-PS and its CC24 controller will run the embedded EPICS server making this solution very "light" and easy to maintain. CSS Operator interfaces are already used to control sets of HV-PS to tune DC electrical fields.

Since 2017, more than 200 HV-PS channels are used at CENBG for the HRS, GPIB and PIPERADE.

Pulse Pattern Generator (PPG)

Ion traps such as PIPERADE need a configurable real-time "conductor". This sequencer also called PPG has been developed for GSI-SHIPTRAP and CERN-ISOLTRAP [16] on RIO FPGA, Real-Time PCI boards with LabVIEW software. We developed seven years ago a PPG for PIPERADE reusing the same LabVIEW FPGA State-Machine on a CompactRIO (NI-7410) to generate time sequences over 32 digital outputs with 10 ns time resolution. High speed LVTLL modules (NI-9402) trigger Agilent function generators (Traps RF excitations) and gate home-made high-speed and high voltage switches (ions injection – transfer - extraction).

The PPG IOC is running on the remote Linux PC to benefit from a "real" EPICS server with full functionalities compared to the uncomplete LabVIEW EPICS Server.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

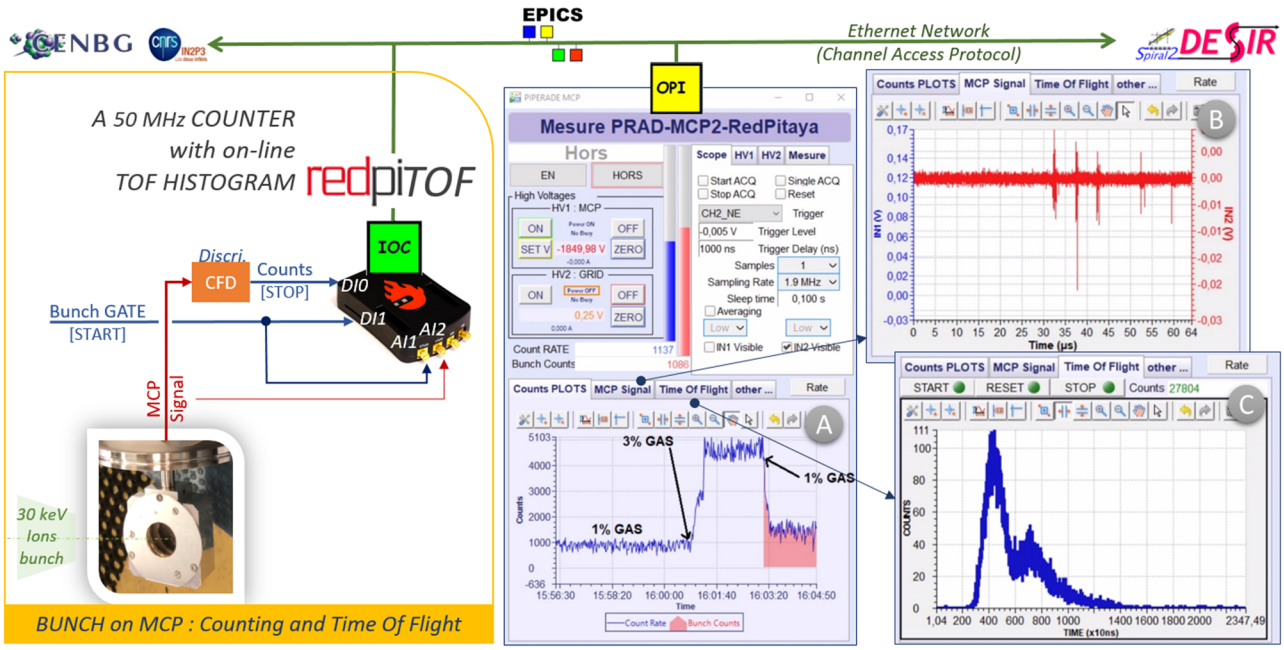


Figure 4: Illustration of our RedpiTOF device developed on a Redpitaya board to count ions detected by a micro-channel-plate (MCP) and measure the Time Of Flight (TOF) with 10 ns time resolution. The TOF histogram (C tab) is constructed on-board and the embedded EPICS IOC allows EPICS Clients to control it. Counting rates (see "A" tab on figure), MCP analogue signal (see "B" tab) and TOF histogram (see "C" tab) are monitored online on the RedpiTOF CSS Operator Interface.

A LabVIEW EPICS Client is running in the CRIo and "feed" the IOC to control the PPG (time sequence definition and start/stop controls) and update the current PPG status with configuration read-back values.

The same PPG device has been cloned to control via five HV fast switches the GPIB buncher time sequence and define the bunch time structure.

Redpitaya for Beam Diagnostics

Bunch intensity on FCup Bunched ion beams produced by the GPIB RFQ-Cooler-Buncher have to be characterised. Hardware solutions are needed to measure the time dispersion, longitudinal emittance and energy dispersion of these microseconds length bunches produced with a slow repetition rate (few bunches per second).

In mid-2019 we chose the cost efficient Redpitaya (RP) board (STEMLab 125-14) to monitor high intensity ion bunches. This choice was motivated by:

- The High performance "dual channel scope" (125 MSamples/s; 14 bits ADC on +/− 1V input signal).
- The board size and SMA connectivity: Easy integration close to beam diagnostics without any hardware development.
- Its Digital Inputs/Outputs usable to drive an amplifier and check its status
- Its Linux operating system (Ubuntu) with the embedded EPICS IOC already developed by the Australian Synchrotron [17].
- The relative low-cost of this complete solution.

A high-speed transimpedance Femto amplifier (DHPCA) amplifies and converts the collected charges of a FCup into a voltage signal. This amplified signal is then injected in one of the two RP SMA analogue input and an operator interface is able to monitor the acquired high-speed signal from the waveform process variable generated in the embedded IOC. The amplifier range is remotely selected using some digital I/O also used to read the amplifier status.

Counting and timing with RedpiTOF Ion counting and TOF measurements using a Micro Channel Plate (MCP) detector are required for PIPERADE trap measurements and GPIB commissioning. Although **c**ounting and coincidence solutions have already been developed on RP boards [18, 19], we started the RedpiTOF development in end-2019 (see Fig. 4).

In addition to the RP scope and wave generator IP, new counting and TOF functionalities using two 100 MHz digital inputs (DI) have been implemented on the Xilinx FPGA and dual core ARM (see Fig. 5). This new "device" is based on a 50 MHz counters (100 MHz clock) and a real-time TDC with 10 ns time resolution implementation.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

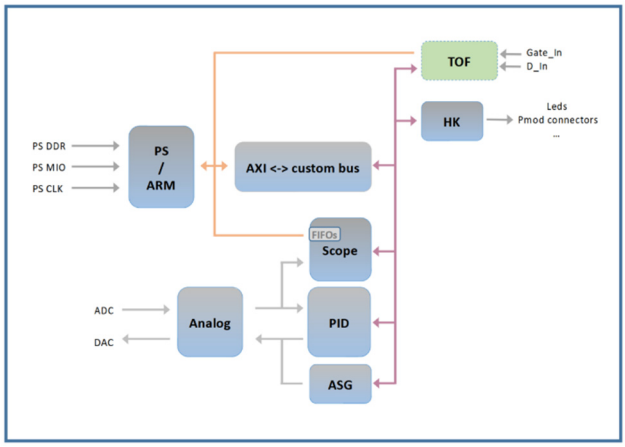


Figure 5: Overview of the firmware architecture. It illustrates how the new TOF function (in green) is interfaced with the original system (in blue).

The first DI is used as a "Gate" signal. The TDC "starts" on a rising edge, while the second DI is "reading" ion signals and act as the TDC "Stop" on rising-edge signal coming from the detector (MCP) discriminated output.

This development has been completed last year with the on-board 16k channels TOF histogram construction, controls (start, stop and reset) and bin size selection (10 ns to 80 ns) to permit TOF dynamic up to 1.3 ms (see Fig. 6).

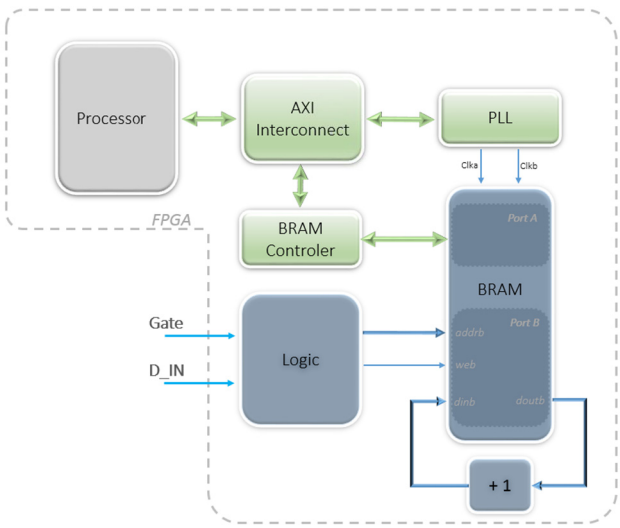


Figure 6: Here is the architecture of the RedpiTOF FPGA implementation. Histogram construction is based on the use of a dual Port RAM. The two digital inputs are processed into the logical module (Logic) to address the first RAM port and increment the value of the selected memory register (Memory address = TOF channel number). The second port is used by the processor to read the histogram over AXI bus with the EPICS driver.

Since January 2021, two RedpiTOF boards are daily used: one for the GPIB bunch characterisation and the other for PIPERADE pennings traps.

CONCLUSION

The Control System (CS) developments we made at CENBG are already used to test and commission the HRS separator, the GPIB RFQ-Cooler-Buncher and PIPERADE penning traps. These slow control CS developments have to be extended to the LS, LT, LHR and LHD DESIR beamlines composed with the same electrostatic devices and diagnostics.

Some CS solutions have to be improved in the following years, in particular for beam diagnostics like low intensity current measurement systems needed for bunched beams and an emittancemeter.

We are also preparing the migration on Phoebus to develop most of the operator interfaces and we will use our institute Gitlab IN2P3 for all of our software developments, up to now using a Subversion repository.

The collaborative work with the GANIL CS team is continuing to prepare as much as possible the DESIR beamlines commissioning at GANIL programmed in 2024.

ACKNOWLEDGEMENTS

This work has also been performed with the helpful contribution of PIPERADE and HRS teams, thanks for their feedback and advice as users of these developments.

The authors would like to thank the JYFLTRAP team, in particular T. Eronen, for providing the PyMassScanner program and advice. The authors also thank GANIL CS & Automation group, more particularly C. Haquin, C.H. Patard and Q. Tura who are more and more available for this collaborative work. Funds to the DESIR-EQUIPEX were allocated by the French Ministry of Higher Education and Research.

REFERENCES

[1] B. Blank *et al.*, "Perspectives for mass spectrometry at the DESIR facility of SPIRAL2," Int. J. Mass Spectrom. 349 (2013) 264, https://doi.org/10.1016/j.ijms.2013.03.006

[2] https://www.ganil-spiral2.eu/scientists/ganil-spiral-2-facilities/experimental-areas/desir/

[3] P. Delahaye *et al.*, "New exotic beams from the SPIRAL1 upgrade," Nucl. Instrum.Methods Phys. Res. B 463 (2020) 339, https://doi.org/10.1016/j.nimb.2019.04.063

[4] https://www.ganil-spiral2.eu/en/

[5] F. Déchery *et al.*, "The Super Separator Spectrometer S3 and the associated detection systems: SIRIUS and LEB-REGLIS3," Nucl. Instrum.Methods Phys. Res. B 376 (2016) 125, https://doi.org/10.1016/j.nimb.2016.02.036

[6] T. Kurtukian-Nieto *et al.*, "SPIRAL2/DESIR high resolution mass separator," Nucl. Instrum. Methods Phys. Res. B 317 (2013) 284–289, https://doi.org/10.1016/j.nimb.2013.07.066

[7] M. Gerbaux *et al.*, "The General Purpose Ion Buncher: a radiofrequency quadrupole cooler-buncher for DESIR at SPIRAL2," in preparation (2021).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV002

[8] P. Ascher *et al.*, PIPERADE: "A double Penning trap for mass separation and mass spectrometry at DESIR/SPIRAL2," NIM A (2021), https://doi.org/10.1016/j.nima.2021.165857

[9] L. Dalesio *et al.*, "The experimental physics and industrial control system architecture: past, present and future," Nucl. Instrum. Methods Phys. Res. A 352 (1994) 179, doi:10.1016/0168-9002(94)91493-1

[10] E. Lécorché *et al.*, "Overview of the GANIL Control Systems for the Different Projects around the Facility," ICALEPCS 2017, Barcelona, Spain. 8-13 October 2017 (2018) 406–410 doi:10.18429/JACoW-ICALEPCS2017-TUPHA016

[11] D. Touchard *et al.*, "Status of the future SPIRAL2 Control System," in Proc. 8th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'10), Saskatoon, Canada, Oct. 2010, paper WEPL006, pp. 38-40.

[12] https://controlssoftware.sns.ornl.gov/css_phoebus/

[13] C. Berthe *et al.*, "Programmable logic controller systems for SPIRAL2," ICALEPCS 2019, N.Y USA. 5-11 October 2019, 1089-1092. doi:10.18429/JACoW-ICALEPCS2019-WEPHA013

[14]C. Jamet *et al.*, "Beam diagnostic overview of the SPIRAL2," in Proc. 10th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC'11), Hamburg, Germany, May 2011, paper TUPD06, pp. 314-316.

[15] T. Eronen *et al.*, "JYFLTRAP: a Penning trap for precision mass spectrometry and isobaric purification," Eur. Phys. J. A 48 (2012).

[16] F. Ziegler *et al.*, "A new PPG based on Labview FPGA," NIM A 679 (2012) 1–6 3.

[17] Australian Synchrotron redpitatya-epics driver- https://github.com/AustralianSynchrotron/redpitaya-epics

[18] I. Bekman *et al.*, "Experience and prospects of real-time signal processing and representation for the beam diagnostics at Cosy," in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 970-972. doi:10.18429/JACoW-ICALEPCS2017-TUPHA213

[19] M. Vretenar, N. Erceg, "Energy-resolved coincidence counting using an FPGA for nuclear lifetime experiments," American Journal of Physics 87, 997 (2019). doi:10.1119/1.5122744

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV003

LASER MEGAJOULE FACILITY OPERATING SOFTWARE OVERVIEW

J.-P. Airiau, V. Denis, H. Durandeau, P. Fourtillan, N. Loustalet, P. Torrent
CEA, Le Barp, France

Abstract

The Laser MegaJoule (LMJ), the French 176-beam laser facility, is located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy on targets, for high energy density physics experiments, including fusion experiments. The first bundle of 8-beams was commissioned in October 2014. By the end of 2021, ten bundles of 8-beams are expected to be fully operational.

Operating software are used to automate, secure and optimize the operations on the LMJ facility. They contribute to the smooth running of the experiment process (from the setup to the results). They are integrated in the maintenance process (from the supply chain to the asset management). They are linked together in order to exchange data and they interact with the control command system. This paper gives an overview of the existing operating software and the lessons learned. It finally explains the incoming works to automate the lifecycle management of elements included in the final optic assembly (replacement, repair, etc.).

INTRODUCTION

The LMJ facility is still in construction. We are currently mixing assembly, operating and maintenance activities at the same time. The progressive increase of operating activities concerns the laser bundles (10 of 22 expected) and the target chamber diagnostics (14 of 33 expected) (see Fig. 1).

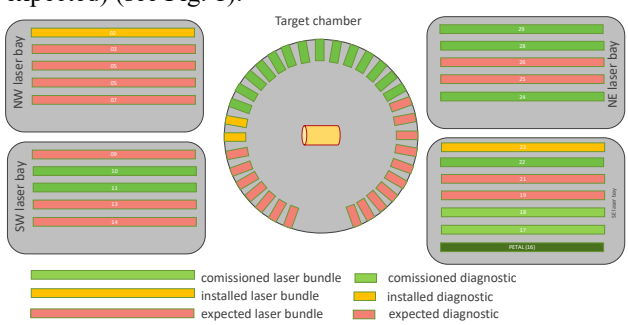


Figure 1: Laser bundles and target diagnostics status.

Laser bundles assembly started in early 2013; the first physics experiment on the facility has been done at the end of 2014 with one laser bundle and two target chamber diagnostics. The end of laser bundle assembly operations is expected for 2025. The full set of target chamber diagnostics is expected later.

Operating software are used to automate, secure and optimize the operations on the LMJ facility. They contribute to the smooth running of the experiment process (from the setup to the results). They are integrated in the maintenance process (from the supply chain to the asset management).

OPERATIONS VS MAINTENANCE

Operating the facility must be still possible while doing assembly and maintenance activities avoiding coactivity issues. These activities must respect safety and security instructions.

Operating such a facility needs to manage many different technical skills (mechanics, electronics, automation, computing and physics). The LMJ is composed of many components. Some of them could be « on the shelves » parts (ex.: pump or motor), others could be specifically designed for the facility and be made of innovative technologies in order to meet the extreme requirements (temperature, pressure and radiation). The supply chain could be very complex for some parts (exclusive know-how, small manufacturing capacity).

The main goal is to fully operate the facility with all the laser bundles and target diagnostic available. But like all complex system, there is sometimes issues on the components (failure, malfunction, etc.). The impact of these issues could be transparent (fault tolerance managed by redundancy), moderated (fault could be bypassed), serious (one laser bundle or one target diagnostic is unavailable) or critical (experiment failure/facility shutdown).

Everything is done to avoid a critical failure and optimize the operation availability. This availability is measure with a Key Performance Indicator (KPI). For example, an availability of 75 % means that quarter of the time the facility could not operate in a nominal way. The time wasted in unplanned maintenance (replacement of faulty parts) downgrade the availability. This KPI is used to measure the technical management of the facility and the ability to anticipate the failures. It is also used for long and mid-term planning.

In order to maximize the availability, it is important to identify all the parts with a Mean Time To Failure (MTTF) lower that the facility's operating life time. It is important to characterize some indicators used to check the "health" status of these parts. By monitoring these indicators, you can anticipate the replacements before the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV003

occurrence of failures. The stock management is based on the MTTF in order to plan the supply of spare parts.

Some of the issues related to the human factor cannot be predicted. It is possible to limit these issues by constantly updating the operating procedure with the feedback (to avoid doing the same error twice). However, it is hard to eliminate totally these kinds of issues.

Despite substantial efforts to reduce unplanned event, you have to deal with operations and maintenance activities. In order to ensure the highest availability KPI, these two activities must be closely linked and complementary. You can't oppose them: operating the facility need maintenance to run and maintenance is useful only if the facility is running (see Fig. 2).

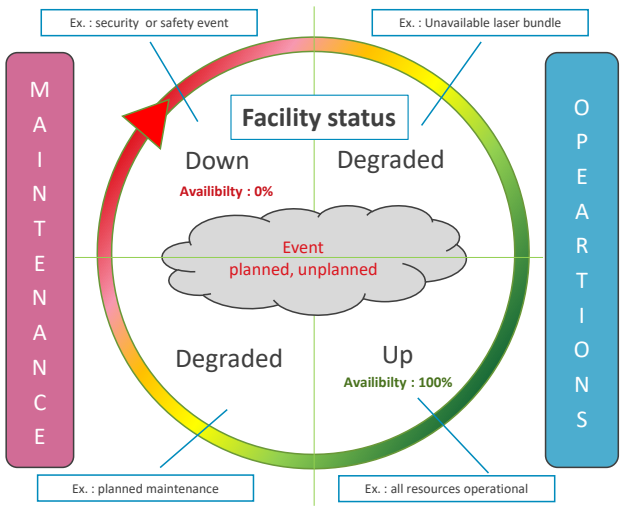


Figure 2: Operations and maintenance duality.

A set of operating software are used for each activity and interfaces between them provide the link.

OPERATING SOFTWARE

Operating software tools are used to control and automate operation and maintenance. These high level tools manage process made of complex tasks in order to reduce the processing time and avoid risk linked to the human factor. Human actions are limited to control critical step in the process (monitoring). The control is made by an operator (technician, contactor) and can be helped by an expert (engineer) if necessary. The idea is to avoid wasting time in low value-added actions and save resources. The tools are used to build dedicated reports to ease the analysis of an issue. The results of the analysis can be re-integrated in the tools and be used to identify a known degraded behavior.

Operating software are used for experiment and maintenance processes.

Operating Software for Experiment

The operating software are developed internally, they are linked with the command control system (see Fig. 3). The shot manager tools (GTIR) is the input and output point of an experiment. The experiment setup is described in GTIR. The results are stored and displayed in this tool. Computing tools are used to:

- predict the control command setup (PARC),
- compute the results and check security and performance during the different step of the shot sequence (rod shot, power shot, etc.),
- calibrate equipment and CC setup with calibration sequence,
- validate experiment results for laser bundles and target diagnostics (PARC + OVID).
- Equipment setup and characteristics are stored in the command control configuration tool (GCI).

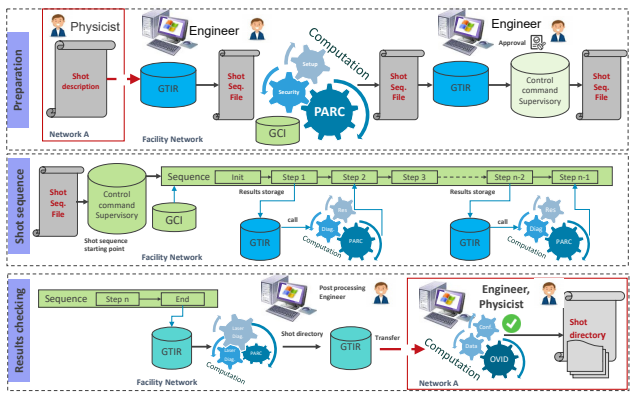


Figure 3: Operating software for experiment.

Operating Software for Maintenance

The facility is composed of more than 200 000 mounting points (registered in the Enterprise Asset Management (EAM) tool). Each mounting point could have an equipment mounted on it. The equipment can be composed of sub parts. Most of the times you have spare parts in the stock. It represents a huge quantity of part and is not humanly possible to identify, check and follow-up the status of every ones. Like for experiment, a set of tools are used for the maintenance activities.

The main tool called GMAO is based on Infor EAM software. It provides a parametrable asset management environment. Every work order is registered in the software in order to follow the facility configuration set-up. Each equipment is identified by a barcode number.

The Electronic Document Management (EDM) called SIROCO is based on Dassault Systems ENOVIA software. Every information related to the lifecycle of equipment (manufacturing, control file, characteristics, etc.) is registered in SIROCO.

The command control configuration tool (GCI) manages the current setup and characteristics used to conduct an experiment.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV003

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The software used to manage the lifecycle of equipment are described on Figure 4. Each new mounting or replacement of equipment is initiated in GMAO as a work order. The planification and the authorization workflow are managed by other operating tools. Once the work permit is authorized and planned, the equipment replacement can be done by an operator helped with the procedure extracted from SIROCO. GMAO and GCI are notified of the part replacement (part with barcode xxx has been replaced with part with barcode yyy). In the case of complex equipment composed of subparts, all the replaceable sub parts are updated in each tool.

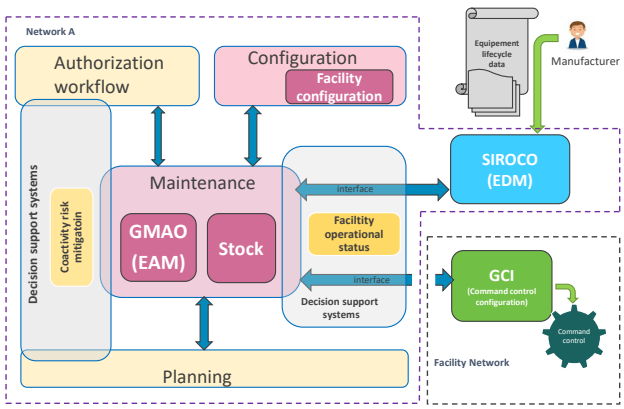


Figure 4: Operating software for maintenance.

A closed link between these three operating software (GMAO/SIROCO/GCI) is necessary.

Interface Between Experiment and Maintenance Software

Characteristics of equipment are supplied by the manufacturer and manage by the EDM tool SIROCO. The facility installed configuration set-up is managed by the EAM tool: GMAO (i.e.: the equipment identified with its barcode is installed and operational). The command control configuration tool GCI registers all the information of all equipment driven by the command control (setup values, characteristics and lifecycle attributes).

The consistency between these 3 tools is critical for two reasons:

- Characteristics from GCI are used for simulation computation (ex. Optics transmission) and equipment calibration (sensibility for an energy sensor). If there is a mismatch in the characteristics associated to equipment (wrong barcode number), you can have biased simulation computation and an important variance on diagnostic results.
- The lifecycle attributes are used to monitor the behavior of equipment during its life in the facility environment and stored in the GCI. These indicators are set by the command control (ex. working time of a pump in hour) or by operating software like PARC [1] and SCIBORG [2] (ex. cumulated energy seen by optics). They are returned to the manufacturer (via SIROCO) when the equipment is down to provide feedback analysis. If there is a mismatch in the indicator associated to equipment, data returned to the manufacturer can be wrong and might bias the results of analysis.

A consistency rule has been specified between tools and software interfaces has been developed to manage the consistency.

LESSONS LEARNED

Network Issues

Maintenance and experiment operating tools are deployed on two physically separated networks. For cybersecurity reasons, the software link is not possible between GMAO/SIROCO and GCI (see Fig. 4). The interface cannot use a synchronous and dedicated protocol to exchange data. The link is done by file exchange between the two networks; each application can load the file in a shared directory and interpret the content.

This one way asynchronous exchange may break the consistency rule. A late update of GMAO data (data transfer issue) can lead to wrong notification of GCI.

The project to move GMAO and SIROCO applications on a network with less cybersecurity constraint has started. The main goal is to update the interface in order to communicate in a synchronous way with web services.

Complex Consistency Rule

The old consistency rule between GMAO and GCI was based on the mounting point key. Equipment and its sub parts (structured as a tree) are referenced in GMAO and geographically positioned with the mounting information. This same information is reported in GCI when the work order to replace the equipment is done. The consistency rule (GMAO vs GCI) works for the equipment only if the tree structure of the equipment and its sub parts are the same in both of the tools.

This hypothesis is not always respected and creates some consistency error.

A new simple consistency rule has been introduced. GCI structure tree uses a Functional Code Number (FCN) as a unique identifier of an element in the tree structure. The FCN information will be added in GMAO and updated after the replacement of the equipment by notification. This rule covers all the use cases.

Identifying Parts on the Facility

Field inventories are conducted to control the “real“ consistency between what is mounted (barcode read on the part by the operator) and the information of GMAO. Sometimes errors are found during these inventories.

The main reason of these errors is due to the difficulty of identifying the right part. The same equipment or part may be found at many places in the laser bay (ex.: autom-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV003

aton). We have optical scanner to identify the part with its barcode but these scanners are not yet connected on mobile devices and the GMAO is not connected to these devices.

A project of deploying local WIFI network in the facility has been initiated. The mobile Infor EAM application will be installed on these devices and provides a direct link to the information of the work order by scanning the bar code.

Spare Parts Inventory

Some parts needs to be replaced or maintained regularly (ex.: pump). After a few year of use, some equipment can be deprecated and the new production series is not compatible with the current setup (ex.: control command hardware). The know-how of a manufacturer can disappear because the product has no more customer (ex.: neodymium doped laser glass for amplifier) so you need to manage a strategical stock for the lifetime of the facility.

The spare parts inventory is about to be completed. For each part, it is important to evaluate the stock size, the maintenance frequency and the maintenance procedure (resources, tools and time expected).

Health Indicators

Some equipment had sensors to monitor their status or performance. These data are processed and stored by the control command supervisory system. Operating tools like PARC [3] manage high level indicator related to laser, alignment and synchronization performances. These health indicators are used to detect degraded operating mode and anticipate maintenance before breakdown.

An exhaustive survey of health indicators is about to start. The main goal is to identify the missing health indicators and the way to implement them.

Global Monitoring System

Each field of activity in the facility has a monitoring system (control command hardware, computing hardware, network, supervision system, software). This could be legacy software (ex. Nagios), logging system or specific software. The monitoring is limited to a specific perimeter and you can't do cross perimeter analysis. In an issue is detected on one of this system, you can't know the impact on the parent systems. Conversely, if you can't run a root causes analysis to find an issue on the child systems.

The project to deploy a global monitoring system on the facility (LMJ Dr.) has been initiated. This system will extract, load and transform data from the different existing monitoring subsystem including health indicators. The data will be stored in a standardized format (data + metadata) as time series.

The primary goal of this system is to be able to manage multipurpose monitoring. For example, it will be possible to correlate the malfunction of a diagnostics with the loss of a network link between this diagnostic and the associated computer.

The secondary goal is to use the time series to detect health indicator decrease and/or predict equipment failure (AI tools based on knowledge database).

USE CASE (FINAL OPTIC ASSEMBLY LIFE CYCLE)

To illustrate how closed is the link between operating tools mentioned before, this use case (see Fig. 5) shows the different steps to manage the lifecycle of an element (vacuum window) of the final optic assembly

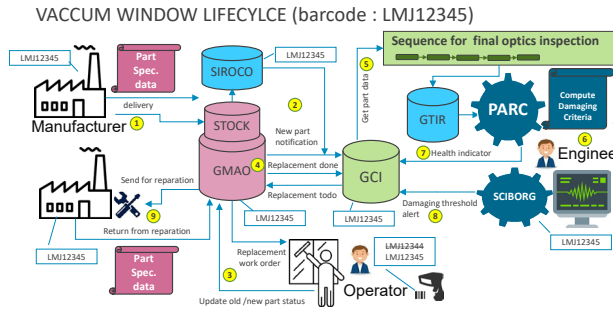


Figure 5: Vacuum window use case.

CONCLUSION

To perform ignition experiments on the facility, the energy expected on target chamber will be very high. The vacuum windows damaging will be important and this part will need to be replaced and repaired several times. The rate of maintenance on this particular optic element will be intensive. The operating software used for experiment and maintenance activities must be perfectly coordinated to ensure this rate of replacement. All the improvements based on the lesson learned needs to be deployed in order to achieve these ambitious goals.

REFERENCES

[1] S. Vermersch, "The Laser Megajoule Facility: The Computational System PARC", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 38-41. doi:10.18429/JACoW-ICALEPCS2015-MOC3O06

[2] J-P. Airiau, V. Denis, P. Fourtillan, C. Lacombe, and S. Vermersch, "SCIBORG: Analyzing and Monitoring LMJ Facility Health and Performance Indicators", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper THCPL04.

[3] J-P. Airiau *et al.*, "PARC: A Computational System in Support of Laser Megajoule Facility Operations", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1034-1037. doi:10.18429/JACoW-ICALEPCS2017-WEAPL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV005

TOWARDS A NEW CONTROL SYSTEM FOR PETRA IV

R. Bacher, T. Delfs, D. Mathes, T. Tempel, T. Wilksen
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

At DESY, an upgrade of the PETRA III synchrotron light source towards a fourth-generation, low emittance machine PETRA IV is currently being actively pursued. The basic concept of the control system of PETRA IV is to exploit synergies between all accelerator facilities operated by DESY. The key figures of PETRA IV's new accelerator control system include the DOOCS control system framework, high-end MTCA.4 technology compliant hardware interfaces for triggered, high-performance applications and hardware interfaces for conventional slow-control applications compliant with industrial process control standards such as OPC UA, and enhanced data acquisition and data storage capabilities. In addition, the suitability of standards for graphical user interfaces based on novel Web application technologies will be investigated. Finally, there is a general focus on improving quality management and quality assurance measures, including proper configuration management, requirements management, bug tracking, software development, and software lifetime management. The paper will report on the current state of development.

INTRODUCTION

With PETRA III, DESY operates one of the best storage ring X-ray radiation sources in the world. PETRA III is a 2300-metre-long storage ring feeding 24 user beamlines. It is operated either in brightness mode (480 equally distributed bunches, 120 mA stored beam) or in timing mode (40 equally distributed bunches, 100 mA stored beam). Research groups from all over the world use the particularly brilliant, intense X-ray light for a variety of experiments - from medical to materials research.

DESY plans to expand it into a high-resolution 3D X-ray microscope for chemical and physical processes. PETRA IV [1] will extend the X-ray view to all length scales, from the atom to millimetres. Researchers can thus analyse processes inside a catalyst, a battery or a microchip under realistic operating conditions and specifically tailor materials with nanostructures. PETRA IV also offers outstanding possibilities and optimal experimental conditions for industry.

PETRA IV will replace the PETRA III facility and will be housed by the existing PETRA III buildings. An additional experimental hall will provide space for additional 18 user beamlines. In addition, a new synchrotron (DESY IV) will serve as booster between the existing electron source LINAC II and PETRA IV.

In 2020, a preparatory phase for the future project PETRA IV was initiated with the aim of submitting a Technical Design Report by mid-2023. Construction work is scheduled to begin in early 2026, followed by a commissioning phase in 2028.

The following chapter will describe the baseline of the accelerator control system of the future PETRA IV facility.

CONTROL SYSTEM DESIGN BASELINE

The development and implementation of the future PETRA IV accelerator control system will be embedded in a long-term process to consolidate the whole accelerator control system landscape at DESY and to take advantage of synergies between the accelerator facilities operated by DESY. The accelerator control system of PETRA IV will closely follow the control system concept implemented at the European XFEL. Consequently, support and maintenance of the existing control system framework used at PETRA III will not be continued beyond its expected lifetime.

Control System Framework

The Distributed Object-Oriented Control System (DOOCS) [2] will form the basis of the future control system of PETRA IV. DOOCS is the established control system framework at FLASH, European XFEL and other conventional accelerator facilities operated by DESY, as well as advanced accelerator projects based on plasma wake field acceleration.

DOOCS is based on a distributed client-server architecture combined with a device-oriented view. Each control system parameter is made accessible via network calls through a device application. Its transportation layer is based on the standardized, industrial RPC protocol. The DOOCS framework is written in C++ and supports a variety of fieldbus and hardware interfaces via device classes. These are accessible through additional libraries which can be linked as needed individually to the server core library. Libraries for creating client applications in C++, Java, Python or MATLAB are available either as a separate implementation or as C-bindings. Through the client API DOOCS provides access to multiple popular control system such as EPICS and TANGO. At DESY, EPICS is used for facility control (electrical power and water distribution, ventilation and air conditioning) and control of the cryogenic systems, while TANGO is the standard control system for operating the beam line components and the experimental equipment.

The initial development of DOOCS dates back to 1993. Since that time, it has steadily developed into a powerful, reliable and versatile control system. Recently, a roadmap was established to meet the increasing user demands over the next decade and to continue to keep pace with the rapid developments in IT technologies.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV005

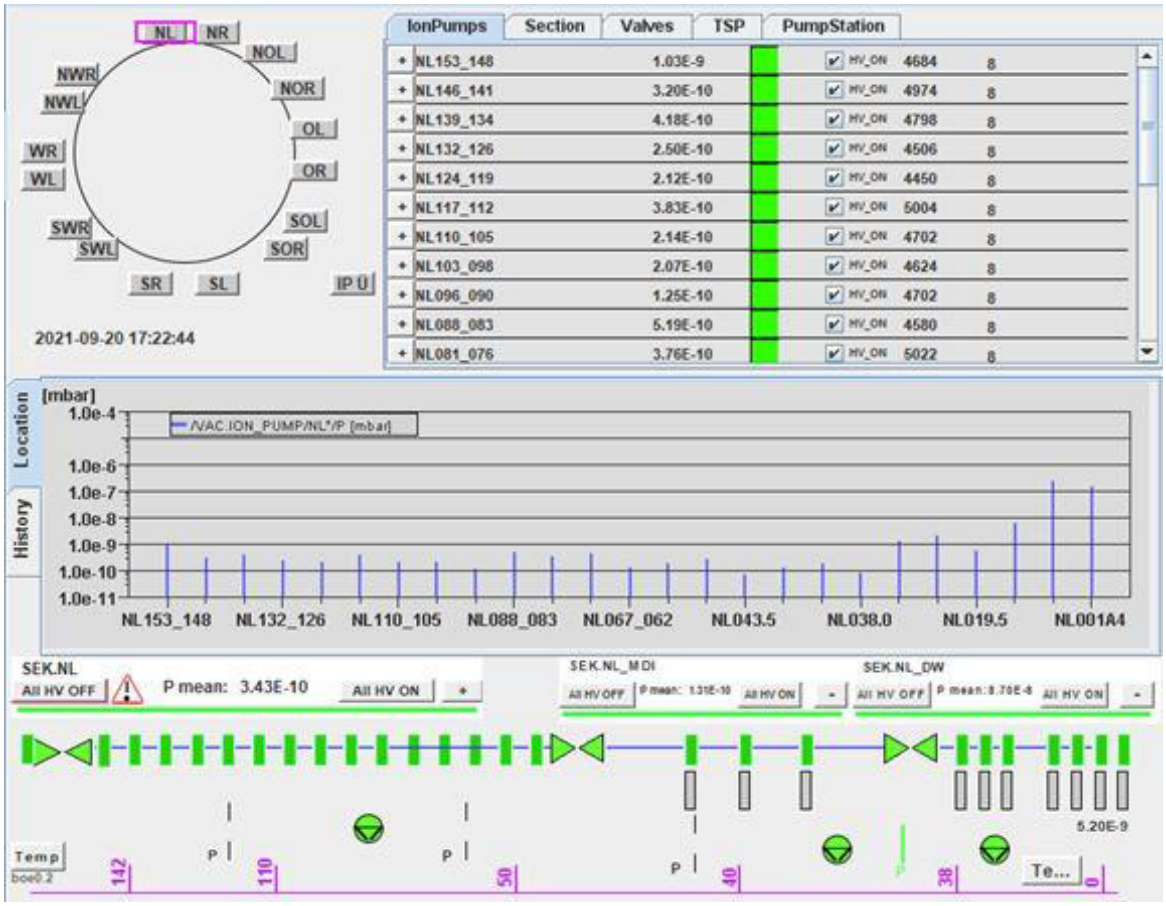


Figure 1: Sample JDDD display.

Graphical User Interfaces

The proven Java DOOCS Data Display (JDDD) [3] is chosen as standard user interface. JDDD is a Java application and follows a thin-client approach with a functional and rich set of widgets. Individual UI components can be easily created through a versatile editor IDE without the knowledge of any programming language (Fig. 1).

While JDDD is the tool of choice for the standard beam operation as well as operating technical accelerator devices and systems, Python is increasingly becoming the preferred programming language for rapid prototyping and visualization of scientific procedures and data.

Even if JDDD also provides a secure, HTML5-based Web interface, with the advent of modern Web standards such as Progressive Web Apps (PWA) other promising alternatives to design graphical user applications seem to be available. PWA are multi-platform, browser-based applications with a look-and-feel of versatile classical desktop applications. The potential of graphical user applications based on the React [4] JavaScript framework is currently investigated.

Hardware Interfaces

In general, the hardware interfaces for triggered, high-performance applications (e.g. beam diagnostics, injection / ejection system, feedback systems, timing / synchronization system, machine protection system, RF control) will be compliant with the high-end MTCA.4 technology [5]. MTCA.4 is the accepted long-term standard for the DESY accelerators and is enjoying growing popularity within the accelerator community and the related industry. The operating system for server hosts running within the MTCA.4 platform will be Linux.

The base configuration of a MTCA system includes a power-supply, a Management-Controller Hub (MCH), a CPU as an Advanced Mezzanine Card (AMC), a Timing System AMC, and optionally a Timing System Rear Transition Module (RTM) if required.

Based on the existing MTCA.4 timing module used at the European XFEL, a successor model is currently being developed at DESY. This module will function either as a transmitter or as a receiver. The module will distribute the RF reference signal, provide low-jitter clocks and trigger signals as well as beam-synchronous data such as a timestamp, revolution counters, beam modes, bunch pattern and bunch current via a dedicated optical fibre network. Storage ring, booster and linac will each be equipped with central timing system components, synchronized across the overall facility. Beamline experiments can make use of the same MTCA-based timing system hardware to exploit all accelerator-provided timing system information.

Dependent on the location in the accelerator, the MTCA systems will be equipped with additional digital I/O AMC modules for specific read-out, measurement and control tasks, e.g. interface boards to the beam position monitor

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

front-end electronics or feedback controllers as well as ADC boards to acquire measured bunch current pulses or HV-pulses generated by the injection and ejection elements.

All of these modules can be managed remotely via Intelligent Platform Management Interface (IPMI) interfaces and the MCH. Linux drivers with hot-swap support allow for PCIe access of the various AMC and RTM modules from applications running on the CPU AMC.

Likewise, the hardware interfaces for conventional slow-control applications (e.g. magnet power supplies, vacuum system) will be compliant with industrial process control standards preferably providing a well-established and widely-used industrial API.

Special emphasis will be put on the OPC UA interface technology [6]. All power converters for magnets as well as power supplies for getter pumps of the vacuum system will implement an OPC UA server. DOOCS provides a generic bridge server, which seamlessly integrates OPC UA devices into the accelerator control system.

Front-end hardware systems based on the Beckhoff controller technology (EtherCAT / OPC UA) will also be used in many cases, e.g. to control the motors of the insertion devices or of the movable girders supporting all accelerator components. In addition, classical PLC system have to be interfaced. In both cases, generic bridge servers to the accelerator control system are available.

Data Acquisition and Archiving

Systems and tools for data acquisition, data archiving and data providing will be implemented with domain-specific interface standards and technologies. Both, time series data as well as snapshot data have to be considered.

Time series data include data from fast data streams in synchronism with the beam revolution frequency (130 kHz) such as single-turn orbit data, and data from slow data stream updated asynchronously with less than 100 Hz. These include e.g. multi-turn orbit data or magnet currents.

Snapshots are measured and stored once and are triggered either by a value change (e.g. in case of a device error), by a specific event (e.g. at beam injection), or by an operator request (e.g. while performing a study).

For both cases, versatile visualization and analysis tools have to be provided. Particular emphasis will be placed on the capability to support data science applications operated in either online (e.g. learning feedbacks, learning tuning procedures) or offline (e.g. failure prediction, predictive maintenance forecast) mode.

High-Level Control Applications

A team of controls experts and accelerator physicists has been already established to interface specific needs of beam commissioning and operations and implement corresponding tools and applications.

The well proven MATLAB Middle Layer library suite [7] supplemented by procedures developed for PETRA III operation will be adapted for further use at PETRA IV. In addition, novel control concepts based on advanced machine learning algorithms are being developed and tested at PETRA III. Resulting applications to be intended afterwards for regular standard operations will be transformed to well-behaved operator panels or server applications.

Similar to the so-called Virtual European XFEL Accelerator [8] a Virtual PETRA Accelerator infrastructure is being set-up. It will be used to test new concepts, enhancements or just modified and improved applications before they will be put into the field which can potentially save significantly commissioning and machine studies time.

Quality Assurance

High availability is a key parameter of modern synchrotron light sources. This must also be reflected in the availability and quality of the software and hardware components of the control system.

Various quality assurance measures are being set-up. These include a revision of the issue and bug tracking workflow and a proper management of requirements for the control system during the preparatory, construction, commissioning and operating phase of PETRA IV. A template has been worked out to document the requirements for the control system which will be regularly reviewed and adapted if needed. Training courses for application developers covering various topics, e.g. application software development, graphical user interface design, software testing etc., are being organized.

The control system of PETRA IV will consist of a huge number of software artefacts. In contrast to PETRA III, also the number of hardware components is significantly larger. A comprehensive configuration management of software and hardware components has to be provided during all stages of the PETRA IV life cycle, e.g. by maintaining a configuration management data base or by well-defined workflows and processes for change and release management.

ACKNOWLEDGEMENTS

The authors thank the PETRA IV project team at DESY for constructive discussions and the Helmholtz Association of German Research Centers and the Federal Ministry for Education and Research for supporting the preparatory phase for the future project PETRA IV.

REFERENCES

[1] PETRA IV Conceptual Design Report, https://bib-pubdb1.desy.de/record/426140/files/DESY-PETRAIV-Conceptual-Design-Report.pdf

[2] DOOCS, https://doocs.desy.de

[3] JDDD, https://jddd.desy.de

[4] React, https://reactjs.org

[5] MTCA.4 Standard, https://www.picmg.org/open-standards/microtca

[6] OPC Foundation, https://opcfoundation.org

[7] MATLAB Middle Layer, https://github.com/atcol-lab/MML

[8] R. Kammering *et al.*, "The Virtual European XFEL Accelerator", in Proc. ICALEPCS 2015, Melbourne, Australia, October 2015, pp. 578-580. doi:10.18429/JACoW-ICALEPCS2015-TUD3004

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV006

THE NEW SMALL WHEEL LOW VOLTAGE POWER SUPPLY DCS FOR THE ATLAS EXPERIMENT

Christos Paraskevopoulos[1,2]
[1]National Technical University of Athens, Zografou, Greece
[2]Brookhaven National Laboratory, Upton, NY, U.S.A

Abstract

A series of upgrades are planned for the LHC accelerator to increase it's instantaneous luminosity to $7.5 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. The luminosity increase drastically impacts the ATLAS trigger and readout data rates. The present ATLAS Small Wheel Muon(SW), will be replaced with a New Small Wheel which is expected to be installed in the ATLAS underground cavern by the end of the Long Shutdown 2 of the LHC. Due to its complexity and expected long-term operation, the New Small Wheel (NSW) requires the development of a sophisticated Detector Control System (DCS). The use of such a system is necessary to allow the detector to function consistently and safely and have a seamless interface to all sub-detectors and the technical infrastructure of the experiment. The central system handles the transition between the probe's possible operating states while ensuring continuous monitoring and archiving of the system's operating parameters. Any abnormality in any subsystem of the detector triggers a signal or alert (alarm), which notifies the user and either defaults to automatic processes or allows manual actions to reset the system to function properly. The part that will be described is the modular system of Low Voltage(LV). Among its core features are remote control, split of radiation sensitive parts from parts that can be housed in a hostile area and compatibility with operation in the radiation and magnetic field as in the ATLAS cavern. The new Low Voltage Intermediate Control Station(ICS) will be used to power the Low Voltage Distributor(LVDB) boards of the NSW and through them, the readout and trigger boards of the system providing data and functioning safely.

ATLAS NEW SMALL WHEEL

ATLAS [1] is the largest high-energy physics detector ever built by man. The LHC delivers millions of collisions each second, that take place in the heart of ATLAS. These collisions though, create a dangerous radiation environment, which the detector has to endure. To efficiently handle the increased luminosity of the High-Luminosity LHC (HL-LHC), the Small Wheel (SW) which is the first station of the ATLAS muon spectrometer end-cap system, will be replaced. The New Small Wheel (NSW) [2] will have to operate in a high background radiation region (up to 22 kHz/cm^2) while reconstructing muon tracks with high precision as well as providing information for the Level-1 trigger. The New Small Wheel consists of two detector technologies of gaseous detectors, the first is called small-strip Thin Gap Chambers (sTGCs), and the second comes from the category of micro-pattern gaseous detectors and is named Micromegas (MMG) [3]. The NSW apart from the two new detector technologies is a platform presenting new Custom ASICs and electronic boards. The readout system is based on Front End LInk eXchange (FELIX) housing 2.5 Million readout channels, common configuration, calibration & Detector Control System (DCS) path and new power supply system.

THE NEW SMALL WHEEL DETECTOR CONTROL SYSTEM

In order to monitor and control the operation of the detector, a framework has been devised, which allows for remote supervision and intervention in the detector and its sub-systems: The Detector Control System (DCS) [4].The DCS is simply a Supervisory Control And Data Acquisition (SCADA) system equipped with User Interfaces (UIs), automated scripts and control/monitor functionality. This control scheme is used daily, in the ATLAS Control Room. It is also used by the subdetector experts to guarantee a safe and efficient physics run. The main task of the DCS is to enable the coherent and safe operation of the full detector by continuously monitoring its operational parameters and its overall state.

Finite State Machine Hierarchy

The chosen software used for the back-end system is called WinCCOA [5]. WinCCOA is a highly modular, device oriented product also with event driven architecture. The back-end system, used by all the four LHC experiments, is implemented using this commercial SCADA package. On top of the SCADA package, the Joint Controls Project (JCOP) framework [6] provides a set of software tools and guidelines and assures the back-end homogeneity over the different subsystems, sub-detectors and LHC experiments. The projects are mapped onto a hierarchy of Finite State Machine (FSM) elements using the JCOP FSM toolkit . The FSM is conceived as an abstract machine that is able to be in only one of a finite number of states at a time. The state can change when initiated by a triggering event or condition (transition state).

System Architecture

The NSW DCS projects closely follow the existing look, feel and command structure of MUON DCS, to facilitate the shifter and expert operations. The top node of both MMG and sTGC will propagate its state and receive commands from the ATLAS overall DCS. Shifters will mainly use the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 1: The overview of the ATLAS MUO DCS FSM structure with NSW DCS integration and brief structure of the sub-detector node structure.

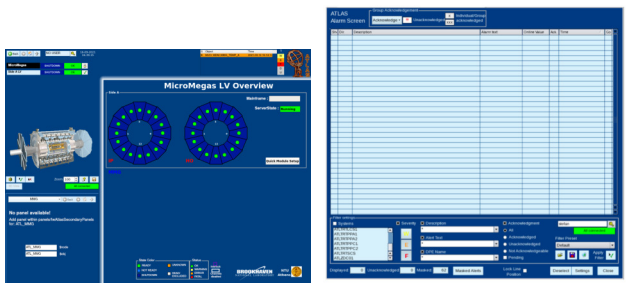


Figure 2: The DCS Overview panel and the Alarm Screen.

Overview DCS FSM panels and DCS Alarm Screen (Fig. 2).

THE LOW VOLTAGE PROJECT

Power Supply

Low voltages for MMG and sTGC are supplied by the widely used CAEN system [7], which is developed for the LHC experiments. Among its core features are:

- Remote control
- Split of radiation sensitive parts from parts that can be housed in a hostile area
- Compatibility with operation under radiation and magnetic fields as in the ATLAS cavern

All the devices are reachable only via the ATLAS Point 1 network. Communication with the control machines is achieved through OPC UA server-client connection. The NSW will be installed at both Side A and C ATLAS End Caps.

The LV CAEN OPC UA servers are also partitioned into Side A and C and deployed on the equivalent host servers. The OPC UA clients, running on the same machines, gather the address space of the individual channel parameters and transmit them to the projects. The two CAEN mainframes of type SY4527 that control the NSW LV also also follow the side architecture of the OPC UA servers and host servers. The mainframes are installed in the ATLAS Service area (USA15). Each mainframe houses 8 branch Controllers.

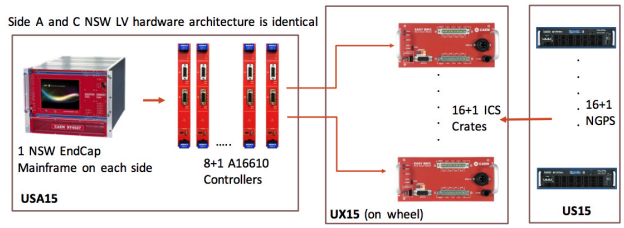


Figure 3: An overview of the installed hardware that is connected to each NSW Endcap.

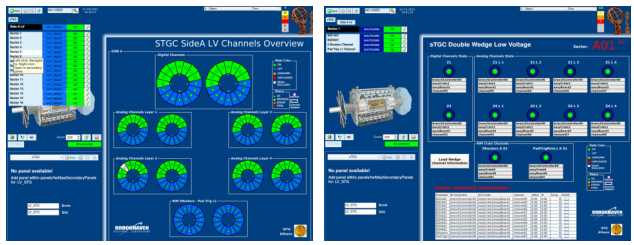


Figure 4: Power supply Control and FSM monitoring panels.

These specifically designed for the NSW branch controllers are connected to the Low Voltage Intermediate Conversion Stage (ICS) modules. ICS will be the provider of the Low Voltage for the readout cards. ICS will be in the hostile environment of the system near to the chambers that are mounted on the NSW.

Each LV controller is connected with two ICS crates, 16 in total for each side. The ICS crates power two adjacent sectors of each sub-detector technology, providing the low voltage for the readout cards. Also two extra ICS are needed for powering the trigger electronics of sTGC detectors. Each Crate hosts 4 modules while every module has 8 output channels, adjusted output channel voltage is 9–11 V and output current 0–17 A.There is also one "virtual" internal channel for general board settings. The LV ICS crates are powered from primary generators called NGPS-30300 with adjustable voltage output. These primary generators are also constructed by CAEN and are located in another service area of the ATLAS experimental cavern (US15). The 34 NGPS primary generators are housed in the allocated NSW racks in US15. The project and CAEN OPC UA server connected to NGPS devices is hosted in a dedicated MUON machine. All the NSW equipment parts were tested and validated by the author with specific automated power cycling and control tests.

Low Voltage FSM

The FSM is based on a strict hierarchical structure with parent-children relations. In this tree-like structure commands propagate from the parents down to the children whereas the states propagate from the children up to the parent on a "most-severe" priority. In this way, when an action is required on all children it is sufficient to give the command to the very top node and correspondingly, the state of the top node summarises the state of all children nodes of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV006

Figure 5: The LV channel node and the ICS Controller accommodating the control panels.

any generation. Three different FSM structures and designs were needed in the case of the NSW LV DCS: the Low Voltage General , NSW NGPS Controller and The ICS module Controller. These three FSM structures are also interfacing through their common hardware parts. The Device Unit (DU) is the base of the FSM and the nodes of this type usually correspond to a device instance and are used to interface to this device. Commands coming from their parents are translated to values, settings of the devices parameters and accordingly, the parameter's values define the node's state which propagates to the highest nodes. A Device Unit node of course cannot contain any children. In the case of the Low Voltage General FSM the DU is a NSW LV individual channel. Each individual channel can be controlled separately while value, status and alarm monitoring is available on the panel. The state and Status of the DU depends on the status data point element of each ICS channel and the communication validity of the OPC UA server.

The design of the ICS Controller FSM DU is based on the channel DU and it is adjusted to accommodate the "virtual" channel error bits. However, the NGPS FSM follows a different design. Each NGPS generator is a DU. This device has different characteristics, functionalities and errors compared to the ICS channel controller. These are seemingly integrated to the control scripts of it.

The projects were validated through daily usage from shifters at the commissioning site as a first stage and then through the NSW control in ATLAS Control room. Except for the FSM design the mapping of channels to detector layers and crates to Sectors was validated. Then the communication with the device setup in ATLAS cavern was established and validated too, The LV FSM operates as expected and is well integrated with the setup in everyday operation.

Interfacing NSW Projects

The NSW LV project will have to interface with the rest of the projects while all are being monitored by the Supervisory Control System.The most important interfacings are the electronics and the temperature monitoring projects. NSW separate configuration/monitor, readout and trigger path consists of 2.4 million readout channels that result in 100 000 DCS parameters. NSW on-detector and cooling temperature monitoring relies on 64 MDT Device Modules (MDM) [8], 32 on each wheel. These devices will monitor the detector

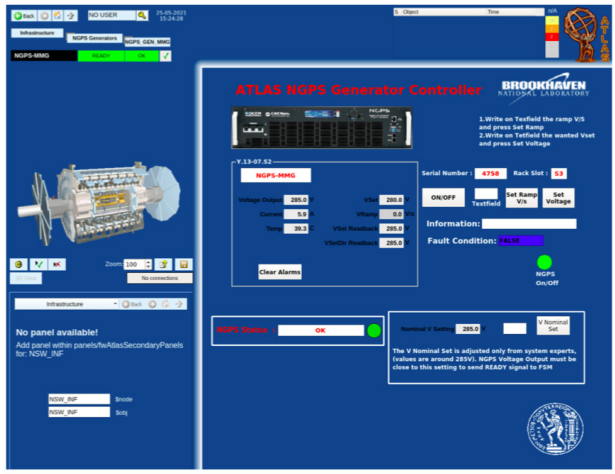


Figure 6: Among others the NGPS DU panel includes: On/Off button, Control of Ramp and Set voltage, Clear Alarms function, Slot and Serial number indications, Vnominal set. Alarms are developed and included to all the aforementioned FSMs.

temperatures and magnetic field. The electronics powering and then the temperature values are directly connected with the LV FSM. In case of alarms incoming from raised temperature value flags, the LV FSM will have to automatically power off the connected devices to avoid failures within the system.

ACKNOWLEDGMENTS

We acknowledge support of this work by the project "DeTAnet: Detector Development and Technologies for High Energy Physics" (MIS 5029538) which is implemented under the action "Reinforcement of the Research and Innovation Infrastructure" funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014-2020) and co-financed by Greece and the European Union (European Regional Development Fund). This work was funded in part by the U. S. Department of Energy, Office of Science, High Energy Physics under Contracts DE-SC0012704, DE-SC0009920.

REFERENCES

[1] The ATLAS Collaboration *et al.*, "The ATLAS Experiment at the CERN Large Hadron Collider", *Journal of Instrumentation*, vol. 3, p. S08003, August 2008. doi:10.1088/1748-0221/3/08/s08003

[2] T. Kawamoto *et al.*, "New Small Wheel Technical Design Report", CERN, Geneva, Switzerland, Rep. CERN-LHCC-2013-006 and Rep. ATLAS-TDR-020, Jun 2013. https://cds.cern.ch/record/1552862?ln=el

[3] T.Alexopoulos *et al.*, "Performance studies of resistive-strip bulk micromegas detectors in view of the ATLAS New Small Wheel upgrade", *Nucl. Instrum. Meth.*, vol. A, no. 937, pp. 125–140, Sep. 2019. doi:10.1016/j.nima.2019.04.050

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV006

[4] A Barriuso Poy *et al.*, "The detector control system of the ATLAS experiment", *Journal of Instrumentation*, vol. 3, p. P05006, May 2008. doi:10.1088/1748-0221/3/05/P05006

[5] WinCCOA, https://www.winccoa.com/.

[6] Oliver Holme *et al.*, "The JCOP framework", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct 2005, paper WE2.1-6O. https://inspirehep.net/literature/699121

[7] CAEN SpA Power Supply systems, https://www.caen.it

[8] Robert G.K. Hart, G. Bobbink, H. Boterenbrood and S. Zimmermann, "The ATLAS MDT Control System", in *Proc. ICALEPCS 2009*, Kobe, Japan, paper TUP079, pp. 263–265.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

THE HV DCS SYSTEM FOR THE NEW SMALL WHEEL UPGRADE OF THE ATLAS EXPERIMENT

E. Karentzos* on behalf of ATLAS Collaboration,
Albert-Ludwigs-University Freiburg

Abstract

The background radiation at the HL-LHC is expected to exceed the one designed for the ATLAS muon spectrometer. In order to cope with the foreseen limitations, the collaboration decided to replace the Small Wheel (SW) with a New SW (NSW) by combining two prototype detectors, the small-strip Thin Gap Chambers or sTGC (STG) & and resistive Micromegas (MMG). Both detector technologies are "aligned" to the ATLAS general baselines for the NSW upgrade project [1], maintaining in such way the excellent performance of the system beyond Run-3. Complementary to the R&D of these detectors, an intuitive control system was of vital importance. The Detector Control System (DCS) for both the MMG and the STG High Voltage (HV), have been developed, following the existing look, feel and command architecture of the other sub-systems. The principal task of the DCS is to enable the coherent and safe operation of the detector by continuously monitoring its operational parameters and its overall state. Both technologies will be installed in ATLAS and will be readout and monitored through the common infrastructure. Aim of this work is the description of the development and implementation of a DCS for the HV system of both technologies.

OVERALL VIEW OF THE NSW HV DCS SYSTEM

Figure 1 illustrates a simplified sketch of the New Small Wheel. The wheel consists of 16 sectors equally divided into large and small sectors. However, the detector geometry is different for each sector type. The numbering of the sectors begins at the left large sector with number 1 and continues clockwise, similarly to the realization in DCS. All large sectors have odd numbers (1,3,5,7,9,11,13,15) while the small ones have even numbers (2,4,6,8,10,12,14,16). Each sector is equipped with the so-called quadruplets of both detector technologies. One quadruplet consists of four layers of a certain detector type.

MMG Scheme

The Micromegas sectors are divided into two quadruplets in radial direction. The quadruplets close to the beam axis defined as xM1, and the outer ones with xM2. Small and large quadruplets just differ in dimension, while similar size layers (xM1, xM2) differs in the number of Printed Circuit Boards (PCB). The larger ones consist of five single readout PCB's which are mounted together to form a detector layer

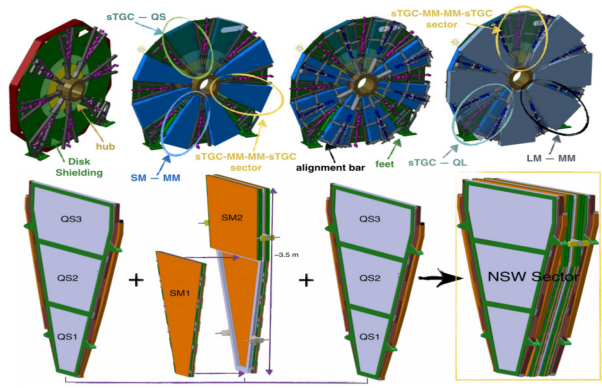


Figure 1: Representation of the NSW (top) with the relevant positions of the NSW sectors, (sTGC and MMG, Small & Large). The different sTGC and MMG modules composing a small sector of the NSW (bottom).

of a quadruplet. On top of the PCB the readout lines are located.

HV Modules Requirements: For both MM quadruplet types a nominal voltage of 570 V/128 µm for the readout is foreseen. The drift boards are supplied by a voltage of 300 V. The current is expected to be less than 1 µA for a readout HV channel. One channel serves one PCB, while the number of channels of all modules sum up to 1024.

sTGC Sheme

The sTGC detectors are divided into 3 quadruplets in the radial direction. The quadruplets naming in radial direction uses the partition definition of inner, middle and outer. A special characteristic is that the innermost quadruplet of the sTGC is divided into two HV sectors in radial direction, due to higher particle flux close at the beam pipe.

HV Modules Requirements: The sTGC quadruplets are operated only with one voltage of around 2.8 kV. There is no aggregation of channels planned, as the number of channels of all modules sum up to 1024. The chosen limits are 1 mA for the innermost HV sector of the inner quadruplets and 0.5 mA for the remaining ones.

Overview of the DCS Internal Structure

In order to incorporate the hardware information into the DCS, several components have been used or developed within the scope of the JCOP Framework [2]. Datapoint Types (DPT) have defined for each hardware component, device and detector segment. Their instances, namely the

* efstathios.karentzos@cern.ch
©2021 CERN for the benefit of the ATLAS Collaboration. CC-BY-4.0 license.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

datapoints (DP), contain all the information (DP Elements) needed in order to describe all the parameters (configs). Additionally, a DP generator developed and programmed to map and set up all the necessary DPs to their associated HV channels., which will be used as the base of the system. In order to translate this structure, to something more meaningful and user friendly, a set of panels has been developed, for the control, monitoring and configuration of the HV system. Finally, a control manager is under development, in order to adjust the high voltage as a function of the pressure and the temperature.

ARCHITECTURE AND HARDWARE REQUIREMENTS

The detailed architecture of the local DCS relies on the structure of the general DCS system of the ATLAS experiment and on the electronics architecture. Each sub-detector is implemented and organized in a unique way but it must also conform to the guidelines defined by the ATLAS central DCS [3, 4]. When special requirements need to be handled by the detector then tailored solutions must be applied.

Architecture

The system architecture consists of several components based on the sub-detector needs and it is supplied by the Universal multichannel CAEN system. The hardware architectures of each sub-system are depicted in Figs. 2 and 3. The MMG chain, the simplest one, consists of 3 SY4527 mainframes [5] that host, supply and control 2 different type of HV boards, A7038ST(P/N, x10 in total) & A7038AP (x16).

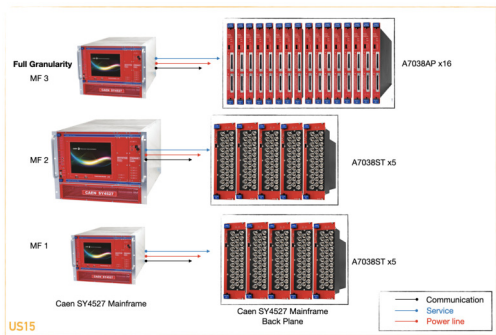


Figure 2: The hardware chain of the MMG as has been installed at ATLAS P1.

The STG chain consists of 2 A1676 branch controllers [6] installed on a SY4527 mainframe, controlling 4 Easy3000s crates [7], hosting 6 HV boards (x3 A3535AP & x3 A3540AP).

Both the easycrates and the easyboards are supplied by two external DC power of 48V generated by separate AC/DC converters. The first generator of type A3485, provides the Service power, needed to control the devices, while the second one of type A3486, is used to supply the line with power. Each generator is controlled by a dedicated branch controller.

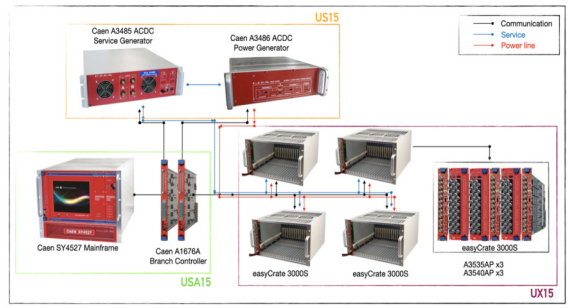


Figure 3: The hardware chain of the STG as has been installed at ATLAS P1.

The communication between the hardware and the DCS is achieved via the OPC UA server-client pairing [8]. Both are deployed on the host servers with the OPC UA clients gathering the address space of the individual channel parameters, transmitting them to the main control stations. The graphical user interface (GUI) of the generators is depicted in Fig. 4, while for the mainframe's monitor and control a dedicated panel is shown in Fig. 5.

Figure 4: Service projects for the monitoring of the 48V ACDC Generators.

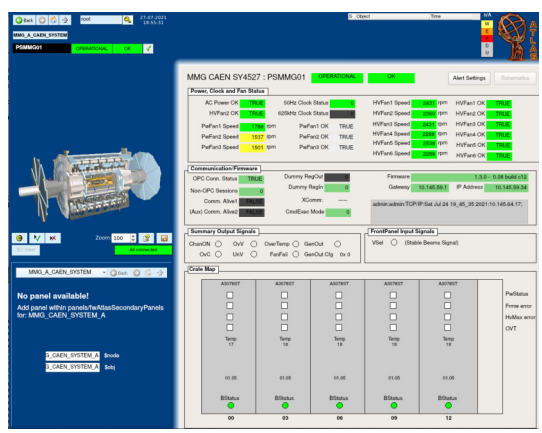


Figure 5: Service projects for the monitoring of the SY4527 mainframes.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

Requirements

The DCS is responsible for the safe and coherent operation of the detector. All ATLAS sub-detectors have their own local DCS, controlling and monitoring the sub-detector and its equipment. Among its core requirements are:

- Remote control, monitoring and configuration of a wide variety of parameters, like voltage and current set/read-back, state and status values along with error flags.
- Active archiving with smoothing and on-the-flight storage in the DCS database [9].
- Data exchange with other sub-systems.
- Discrimination between radiation sensitive and hard parts,
- Compatibility with operation under radiation and magnetic field as in the ATLAS cavern.
- Automated mechanisms, such as the safety Interlock, the Reset Trip, the HV transition when the Beam for Physics is expected or not, etc.
- Communication is established via the OPC UA server-client pairing.

FSM HIERARCHY AND LOGIC

As a detector oriented hierarchy the control system is represented by an homonymous Finite State Machine (FSM) [4] node (MMG/STG). The top node defined as the main control unit, and from there commands are propagated downwards into the device or logical elements. Such units, groups and geographical detector segments are represented by device-oriented (SMI) FSM objects. The current condition of each object is determined by a set of states alongside with their transitions. A state transition is triggered either by a condition change or by a dedicated action. Figure 6 depicts the device and logical state diagrams. It allows a device to change from its ground state OFF to its operational state ON via optional stages, while for logical objects, the mandatory states are READY or NOT_READY, reflecting conditions for which data taking is possible or not. The state UNKNOWN is used when the condition cannot be verified for both cases.

Figure 7 shows an instance of the MMG FSM tree along with each commands. The actual state is determined by the states of the associated children via state rules implemented using SML and they propagate upwards. Custom-made actions can be defined easily via the read control scripts, i.e. interlocking or resetting the flag of a channel, complementing the system.

OPERATION PANELS

Both projects have been developed, following the existing look, feel and command architecture of the other Muon subsystems, in order to facilitate the shifter/expert operations.

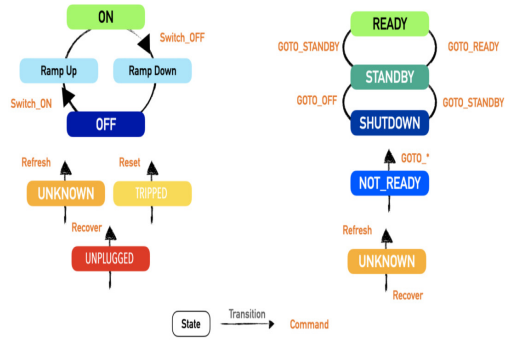


Figure 6: Device (left) and Logical (right) object state model.

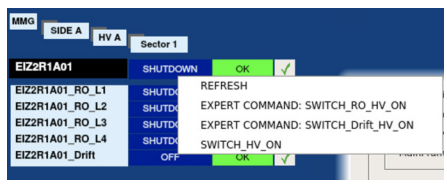


Figure 7: MMG FSM Tree instance at chamber level.

For each individual layer, a main panel has been developed, providing the user with useful information, reflecting the state and status of the detector. All the vitals of each layer can be displayed in trend-plots or/and tables; while a secondary panel provides supplementary to the main panel details.

A general review of the graphical user interface of the top FSM nodes and their constituents are shown in the Figs. 8–13. To facilitate the experts' work also advanced panels and colormaps of the system have been developed. A typical example of a colormap, used by the MMG, is shown in Fig. 13.

Side View

The shifter/expert in the Atlas Control Room usually will have an overview of the sub-system. The typical view for the MMG technology is shown in Fig. 8, where 10 wheels are displayed. As the number of HV sectors within the MMG quadruplets is identical to the partitioning of the quadruplets, their representation in the panel can be retained. The sTGC sectors, seen in Fig. 9, have 3 quadruplets, but 4 HV sectors in radial direction. Therefore, 4 indicators are shown for each sector of the sTGC. Each wheel represents the quadruplets, either close to the IP or HO side. The color indication of the trapezoidal and its accompanied circle refer to the FSM State and Status of the complete quadruplet., respectively. Furthermore, both shapes feature different actions depending on left-, right- or double-clicking [10, 11].

Sector View

The dedicated to the sector panel is shown in Fig. 10. On the top the chosen sector is indicated in a text field. A table is displayed, where all the key parameters for each layer of the quadruplets of the chosen sector are shown. These are the name and location of the connected HV board and channel

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

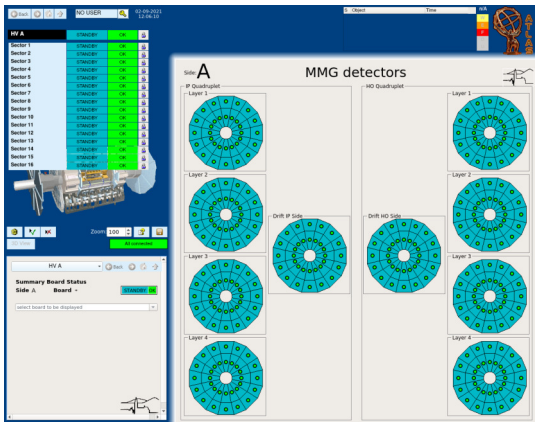


Figure 8: Operational Panels for the MMG sub-system.

Figure 9: Operational Panels for the STG sub-system.

as well as the actual values of voltage and current, accompanied by their error flags. Below the table the connected HV modules are shown, together with additional information of the type, the connection pattern, the temperature, the serial number and the error flags. Complementary to that, a manipulation section on the configuration has been added for convenience during the commissioning at P1 [10, 11].

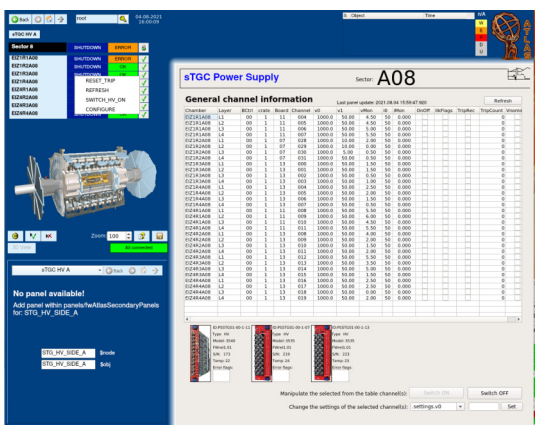


Figure 10: Operational Panels for both MMG and STG subsystems.

Chamber View

In case that a more detailed view of a quadruplet is desired, the panels in Figs. 11 and 12 can be called.In the top the name of the quadruplet is given. Below the name of the quadruplet the names of the connected HV modules are displayed, including their periphery of mainframes, branch controllers, etc. On the right side of the names there are buttons which allows the user to navigate to the corresponding panel, which holds the information about the selected module. Below the fields for the HV modules a table shows more details about the connected channels, their set and monitored voltage or current. The last column of the table shows the errors and has a color-changing indication for fast reading. On each cell of the table, the right-click functionality is enabled. This triggers the instantiation of a trend plot which pops below the table. The trend plot is automatically updated if a change in the values occur. In contrast to this, the table is only updated if the "refresh" button is used. Furthermore, a so called "expert mode" is implemented, followed by the chamber overview panel. This allows shift experts to modify the values of voltage or current of the displayed channels [10, 11].

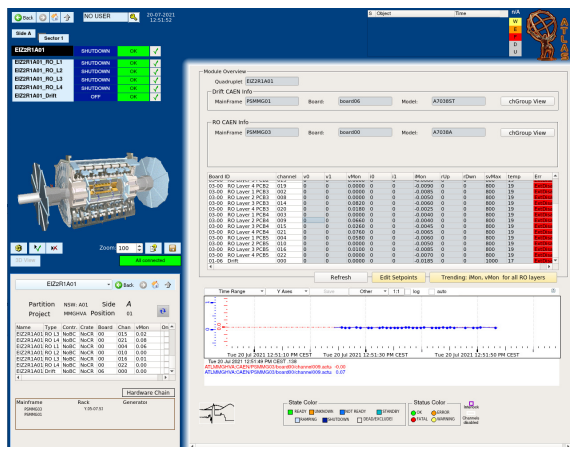


Figure 11: Operational Panels for both MMG and STG subsystems.

Layer View

User can navigate lower to the hierarchy, to the so-called Layer View. Here is the main difference between the p2 rojects. For the MMG project, the layer belong to the family of the LUs while for the STG belong to the DUs. Thus, it will be presented at the next subsection. Figure 12 depicts the layer view of the MMG. There the overview picture can be taken for the all the layer's PCBs, along with its connected high voltage channels. On the right, the key parameters of the layer's operation is given, along with their associated hardware components.

Channel View

The lowest layer of the FSM hierarchy is the so called, Channel or Layer View, depicted in Fig. 13, for the MMG

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

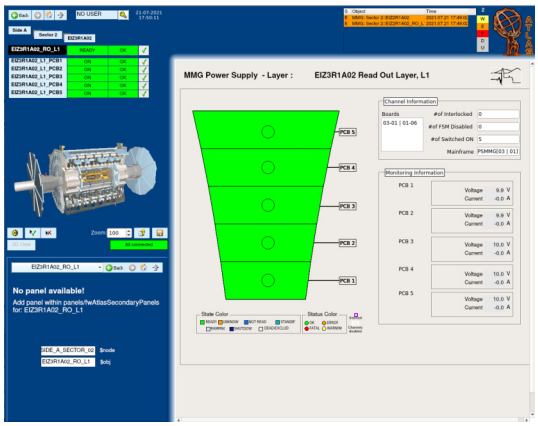


Figure 12: Operational Panels for both MMG and STG subsystems.

and the STG, respectively. There the connected high voltage channel and its model name is displayed in the top. The table contains the key parameters of operation and offers the same functionality as for the quadruplet module view; a trend plot of a variable is called by a right-click in the corresponding cell [10, 11].

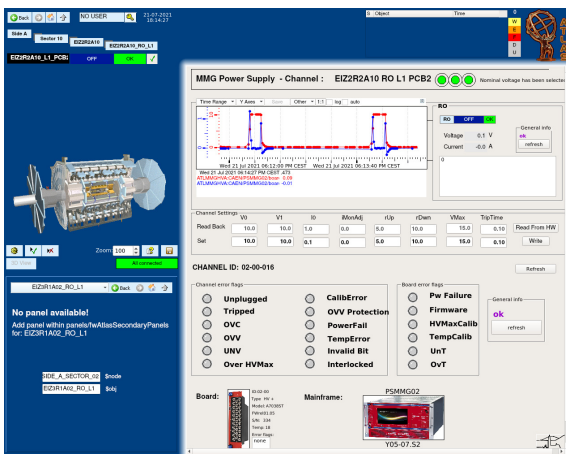


Figure 13: Operational Panels for both MMG and STG subsystems.

Expert Panels

A variety of expert panels (archiving and alarm handling, configuration and check of the periphery address,overall system review, database status, and more) has been developed for both projects, to accommodate all the needs for all type of users (shifter, developer, maintainer, expert) but mainly to facilitate the work of the expert during intervention or debugging and troubleshooting. A typical example of an expert panel, is the colormap, as can be seen in Fig. 14 [10,11]. This panel is developed to provide the controllers a quick overview on the full detector by using color indication. Therefore, 2 wheels are displayed to represent the drift lines for both HO and IP sides. The readout channel are displayed in a colormap matrix (row: sector number, column: pcb number) as colorboxes. Via the drop-down menu it is possible to select the layer and the parameter, which should be displayed. These can be parameters of the HV channels as the monitored or set voltages, currents or the environmental parameters of the quadruplets. The color palette is adjusted automatically to the lowest (blue) and highest (red) value, but also can be set by the operator. This allows the operator to determine a layer which is out of the desired range. By hovering the mouse over a layer a tool tip with the numeric value of this layer is displayed. Additionally, by left-clicking on a rectangle a summary of all of its basic parameters is displayed.

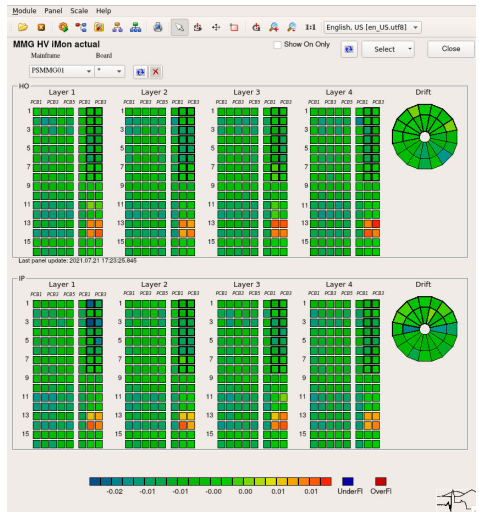


Figure 14: Operational Panels for both MMG and STG subsystems.

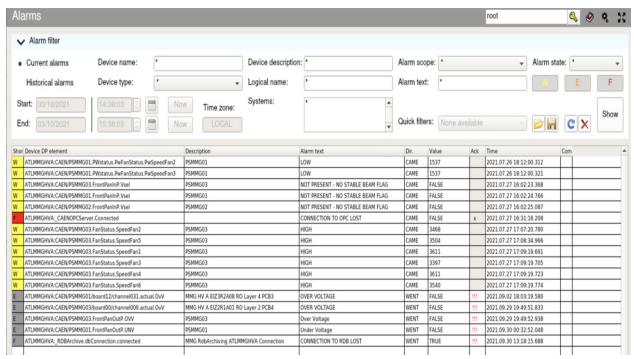


Figure 15: The Alarm Screen.

ALARM SCHEME

A powerful alert system (Fig. 15) notifies the operators about detected anomalies/malfunctions of the system during detector operation; providing higher (FSM) and lower (device) level error signalization. Error signalization within the FSM hierarchy is performed using a parallel tree of "Status" objects. For each FSM object, a dedicated Status instance is assigned with the states OK, WARNING, ERROR, and FATAL, signaling a problem in the corresponding part of the detector attributed to one of the following severity: Warn-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV009

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ing, Error, Fatal. An embedded help system also provides instructions required for recovery.

REFERENCES

[1] T. Kawamoto et al., "New Small Wheel Technical Design Report", CERN, Geneva, Switzerland, Rep. CERN-LHCC-2013-006, Rep. ATLAS-TDR-020, June 2013. https://cds.cern.ch/record/1552862

[2] JCOP Framework https://jcop.web.cern.ch/jcop-framework/.

[3] K. Lantzsch *et al.*, "The ATLAS detector control system", *Journal of Physics: Conference Series*, vol. 396, p. 012028, 2012. doi:10.1088/1742-6596/396/1/012028

[4] Barriuso-Poy, Alex, "Hierarchical Control of the ATLAS Experiment", CERN, Geneva, Switzerland, 2007. https://cds.cern.ch/record/1034400

[5] CAEN User's Manual, "Universal Multichannel power supply system, CAEN Mod.SY1527", Rev. 18, NPO: 00103/97:1527y. Web reference. https://www.caen.it/products/sy4527/

[6] CAEN User's Manual, "Technical Information Manual, CAEN A1676 EASY Branch Controller", CAEN Mod.A1676A, 2015. Web reference. https://www.caen.it/products/a1676a/

[7] CAEN User's Manual, "Technical Information Manual, EMBEDDED ASSEMBLY POWER SUPPLY SYSTEM, CAEN Mod.Easy300/4000", Rev. 16, 5 May 2014, NPO:00105/03:A1676A. Web reference. https://www.manualslib.com/manual/1601885/Caen-Easy3000.html

[8] W. Mahnke and S. H. Leitner, "OPC Unified Architecture: The future standard for communication and information modeling in automation", *ABB Review*, vol. 3, p. 56, Mar. 2009.

[9] Monica Verducci, "ATLAS Conditions Database Experience With the Cool Project", in *10th ICATPP Conference on Astroparticle, Particle, Space Physics, Detectors and Medical Physics Applications*, Villa Olmo, Como, Italy, Oct. 2007, pp. 824–828. doi:10.1142/9789812819093_0137

[10] E. Karentzos, "Research and Development of the Micromegas Detector for the New Small Wheel upgrade in the ATLAS Experiment", 2019, Rep. CERN-THESIS-2019-094, Ph.D. thesis, Natl. Tech. U., Athens, 2019. https://cds.cern.ch/record/2683712.

[11] Thorwald Klapdor-Kleingrothaus, "Gas Gain Studies for MicroMegas Detectors and Development of the High Voltage Control System for the Detectors of the ATLAS New Small Wheel Upgrade", Ph.D. thesis, Freiburg U., Germany, 2019. doi:10.6094/UNIFR/17485

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV010

WORKING UNDER PANDEMIC CONDITIONS: CONTACT TRACING MEETS TECHNOLOGY

E. Blanco Viñuela*, T. Previero, E. Matli, B. Copy, S. Danzeca,
R. Sierra, R. Losito, Ch. Delamare, A. Masi
CERN, Geneva, Switzerland

Abstract

COVID-19 has dramatically transformed our working practices with a big change to a teleworking model for many people. There are however many essential activities requiring personnel on site. In order to minimise the risks for its personnel CERN decided to take every measure possible, including internal contact tracing by the CERN medical service. They performed manual procedures which relied on people's ability to remember past encounters. To improve this situation and minimise the number of employees who would need to be quarantined, CERN approved the design of a specific device: the Proximeter. The project goal was to design a wearable device, built in a partnership with industry (*Terabee* and CERN), fulfilling the contact tracing needs of the CERN medical service. The proximeter records other devices in close proximity and reports the encounters to a cloud-based system. The service came to operation early 2021 and more than 8000 devices were distributed to CERN members of personnel. This publication reports on the service offered, with emphasis on the overall workflow of the project under exceptional conditions and the implications data privacy imposed on the design of the software application.

INTRODUCTION

COVID-19 has dramatically modified working conditions hence affected largely to global economies and enterprise businesses. However, during the pandemic situation essential activities still required personnel on site. In order to keep production stability but ensuring workers safety there was an increased focus on finding solutions to comply with COVID-19 social distancing recommendations [1].

Initially, most of CERN activities adapted and were executed remotely (teleworking), but still some critical activities required work on site. The Health, Safety and Environmental (HSE) unit at CERN introduced a series of measures following the recommendations issued by the Host States. Social distancing and contact tracing have been some of the measures implemented. Contact tracing is defined as the capability of identifying persons who have been in the vicinity of an infected person so their isolation would avoid the spread of the virus. This tracing process was manually conducted by the medical service and required a large effort to keep up with the rate of infection. The process was mostly relying on memories of the infected individuals who should remember with whom they may be in close contact hence does not guarantee perfect traceability.

Two main challenges appeared: (1) help individuals in keeping a **social distance** and (2) timely **identify individuals** who were in close contact with an infected person, the so-called contact tracing.

To tackle the mentioned challenges CERN approved the design of a specific device: the proximeter. The overall project goal was to design a wearable device able to record other devices in close proximity. Additionally an application provides only the close contacts under the request of the medical service.

This publication reports on the service offered, with emphasis on the overall workflow of the project under exceptional conditions and the implications data privacy imposed on the design of the overall service.

THE PROXIMETER

First of all, the definition of two essential terms representing key events widely used during this publication is introduced here: **encounter** as the event of two individuals being within 2 meters distance for more than 30 seconds and **close contact** as the event of an encounter lasting more than 15 minutes which must be traced if required by the medical service.

The device was created with two main objectives: (1) **warning**: so the users are warned once they are not keeping the expected social distancing (2) **contact tracing**: timely provide potential close contacts in case of a positive appears.

Several similar devices with these characteristics were available on the market, but none presenting at the same time the desired precision, scalability to 10000 and more devices, timely data availability and more importantly adaptability in order to protect the sensible data collected at all moments of use and transfer. CERN decided then to issue a competitive tender with those specific requirements.

The company that was awarded the contract, *Terabee* [2], adapted one of its products, the *Terabee mobile robotics positioning system*, to comply with the specifications. In particular, for the connection to the CERN infrastructure *Terabee* implemented the CERN's *miniIoT* (Internet of Things) technology, under licence from CERN, while using its own technology for the encounters tracking based on the accurate Ultra Wide Band (UWB) radiofrequency.

Then, *Terabee* engineered a device that CERN called the **proximeter** (Fig. 1) which become a commercialised wearable product in *Terabee*'s portfolio [3].

* Enrique.Blanco@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

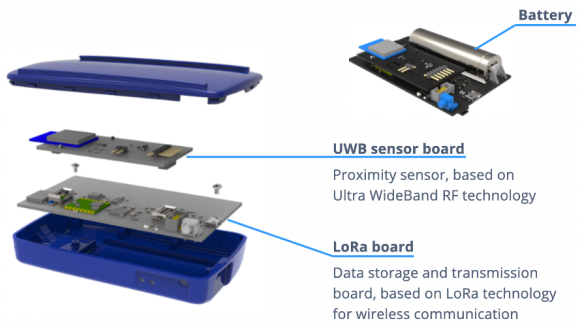


Figure 1: The proximeter device.

Some of the key requirements of the product and the solutions provided are shown in Table 1.

Table 1: Main Requirements and Solutions Given

Requirements	Solutions
Precise detection	UWB technology
Warning features	Buzzer and visual LEDs
Easy to use	Wearable device
Autonomy	Large capacity battery
Encounters accessibility	IoT networking: LoRaWAN®
Privacy	Encryption and no geotagging

The proximeter only knows where it is with respect to other proximeters. It does, however, know that specific information to an accuracy far better than that of mobile-phone-based apps, which makes it very good at telling its holders when they are getting too close, while keeping their whereabouts confidential.

The proximeter device [4] is a twofold design (Fig. 1): it is composed of a mainboard and a sensor-board. The sensor-board hosts a Ultra Wide Band (UWB) sensor. The sensor-board communicates with the mainboard via an SPI bus, that is shared with the LoRa® transceiver. The board is powered by a rechargeable Li-Ion battery rated at 3350mAh. The device also provides a red LED, which is used to assess the charging state, an RGB status LED, and a vibrator motor, which warns the user in case of an encounter is detected.

When the device is powered on, it joins the LoRaWAN® network and sends encounters that were not transmitted, if present. At the end of each encounter, the recorded information is saved in a flash memory and is used to compose the LoRaWAN® packets to be transmitted. Every 15 minutes, the device checks the encounters that have not yet been sent and empties the queue via LoRaWAN®. The communication between the device and the LoRaWAN® network is encrypted via AES-128.

THE IoT INFRASTRUCTURE

As already introduced the device uses LoRaWAN® to exchange the encounters. The LoRaWAN® specification [5] is a Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated 'things' to local or global networks, and targets key Internet of Things (IoT) requirements.

LoRaWAN® is the default IoT protocol for CERN [6] and it has been used in other projects in the last years. The choice of the proximeter for using this network was made in base of the experience already acquired with other devices, the existence of the *miniIoT* platform developed at CERN and the important requirements of coverage, power consumption and confidentiality. The network offers extremely **long-range data** links which allows to exchange data all around the campus (around 60 km2) with minimal deployment. The proximeters are battery-powered and LoRa has a reduction in **power consumption** by a factor of 10 compared to Wi-Fi. LoRaWAN® parameters are only known by the LoRaWAN® network administrators, not even the owner of the device knows them, and all communications are **encrypted** from the device up to the application.

The CERN IT department offers the infrastructure of the network together with a complete architecture which allows to host the device transmitted data.

THE SERVICE

CERN imposed the use of the proximeters as part of the efforts to respond to the challenges posed by COVID-19. To date, more than 8000 people received the wearable device with a goal to minimize potentially harmful encounters and provide contact tracing information in case of need.

Setting up such global service was a non trivial task as the service must take care of different aspects as distribution, asset management, support, data collection and, finally, provide the relevant close contacts to the medical service. All these must be done under strong requirements on personnel safety and privacy. To succeed in this task a diverse team of people in different technical and organisational fields was assembled.

Distribution

Distributing such amount of devices as fast as possible during the pandemic situation constituted the first challenge. Put in practice this demand required a fast and organised distribution mechanism which guaranteed the users safety as crowds had to be avoided.

A dedicated web-based appointment mechanism was set up. Users could reserve a slot to pick up their proximeters at their convenience (location and time). This ensured the safety for the team delivering the devices and the users themselves.

The response of the CERN personnel collecting the devices was extremely good (Fig. 2) considering that the majority of them were teleworking from their homes. This mechanism gave additional trust which indeed allowed a general recall campaign to deal with the release of a newer firmware version (mid February 2021) improving safety features deployed on the first firmware version. The distribution started with a pilot of 950 devices in December 2020 which

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV010

confirmed their effectiveness, successfully reporting violations of the two-metre rule with an accuracy of up to 90%. The full scale roll out officially started early 2021 and the use of the proximeter became mandatory from 1st of March 2021 when the number of 5000 devices distributed was successfully reached.

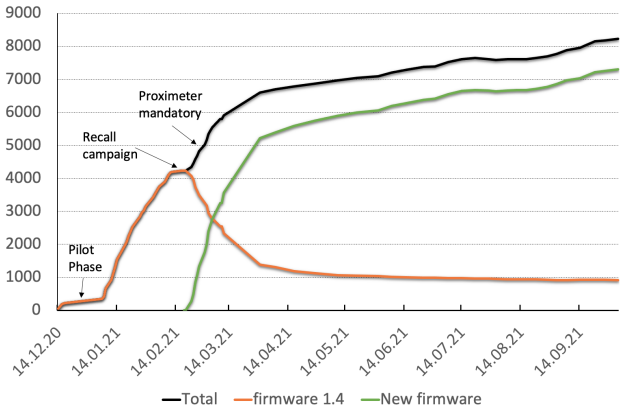


Figure 2: Distribution of the proximeters.

The distribution of the devices was executed by the registration service initially in different sites (e.g. experimental areas, restaurants...) and then centralised in a single location. Additional staff was required to deliver such amount of devices in parallel.

Asset Management

This activity started with the procurement of the devices themselves which were configured with the required encounter parameters (distance: 2 meters and time: 30 seconds) at the factory (*Terabee*). The CERN procurement and supply chain teams worked together to handle the equipment providing them to the distribution point.

All the devices were registered in the computerised maintenance system available at CERN: *InforEAM*®. This allowed proper asset management to cope with the administration of spares and to handle the dysfunctional or lost devices while also playing an essential role on the data protection as will be explained later. The support of this activity was done by the asset and maintenance management team.

Proximeters Support

The success of the project resided on the adoption of the wearable device by the users.

The device is not complex to use as it only requires to switch it on. However users need, both, to understand the different status modes and, more importantly, what and how the personal data shared is exploited. To cope with the first need, a complete user manual was created, distributed with the device and also made available online. To respond to the second need, a comprehensive online training course was created on the learning hub at CERN. This became essential in explaining the goals of the project and showing the kind of data collected together with their potential use by the medical service providing total transparency on personal data use. Moreover the information about the proximeter project was regularly reported to the CERN personnel.

The support of the users was made by a team of experts on different domains (e.g. electronics, software, architecture, data protection...). All support was centralised in the CERN ticketing support service: *ServiceNow*® where a functional element was created to provide the CERN-wide support for the proximeter device. Diverse technical teams were involved in this support. The proximeter functional element provided a central point of support with links to a comprehensive set of the project information, online courses, frequently asked questions, manuals and the privacy notice.

During the current year (project activity) and until mid-September, there have been 216 incident requests and 131 general requests. These figures are considered to be low with respect to the number of deployed proximeters and they corroborate the quality and the ability of the device to be operational.

Data Collection

The proximeter stores the encounters locally with the following data (see example of Table 2): Peer device (*RefTag ID*), own device (*myTag ID*) starting time of the encounter (*Date*) and duration (*NUM*). The last representing the total number of 30 seconds slots of the whole duration of the encounter. Data is regularly sent via the LoRaWAN® network and then stored in a database as records.

Table 2: Data Records Transmitted and Archived

RefTag ID	myTag ID	NUM	Date
24500	12893	32	01-MAR-21 10:26:00
22382	12893	3	01-MAR-21 10:51:00
53019	12893	7	01-MAR-21 10:58:00

The data is kept for a maximum of 14 days which is the time window when the tracing of close contacts may be of interest, then it is deleted. The IDs serve to identify the proximeter involved in the encounter. By design, the proximeters can not register the *geolocation* of the devices, so user's whereabouts remain confidential.

Tracing Exercise

The ultimate phase is the ability of the medical service to trace people in close contact with a tested positive case or a person showing disease symptoms. The medical service is the only service allowed to perform this activity.

A complete software suite, the *proximeter portal* was designed. It allowed the medical service to query the records with the name of a person and to get back the close contacts list. This software suite represents the core of the service as it provides not only this feature but also handles the distribution and management of the devices by the registration service. Figure 3 shows the functionality given when tracing people. The example shows the identification of a close contact with

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV010

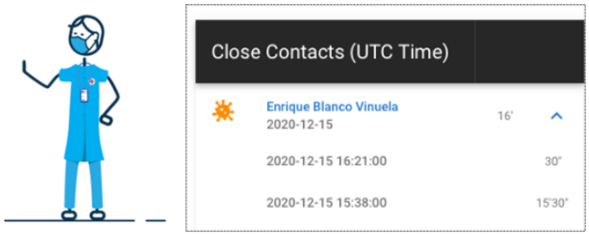


Figure 3: Close contacts from a contact tracing request.

two encounters of 30 seconds and 15 minutes and 30 seconds respectively.

The software suite will be technically introduced later but its functionality appears along this publication in many aspects: e.g. distribution, data privacy, and user interface allowing CERN personnel to consult their own generated data.

THE DATA PRIVACY

CERN collects and uses **Personal Data** related to people who interact with the organisation in carrying out the mission. CERN is committed to respecting the security and confidentiality of the Personal Data for which it is responsible, in accordance with its Data Privacy Protection Policy[1]. CERN processes only such Personal Data as is required for the proper functioning of the Organization. A privacy notice for the proximeter device was created.

Data privacy has been the central concern on the proximeter project, therefore the design of the service including the tools, architecture and the software providing the main functionalities (i.e. assignation, asset management and tracing) have followed strong principles to protect the users privacy.

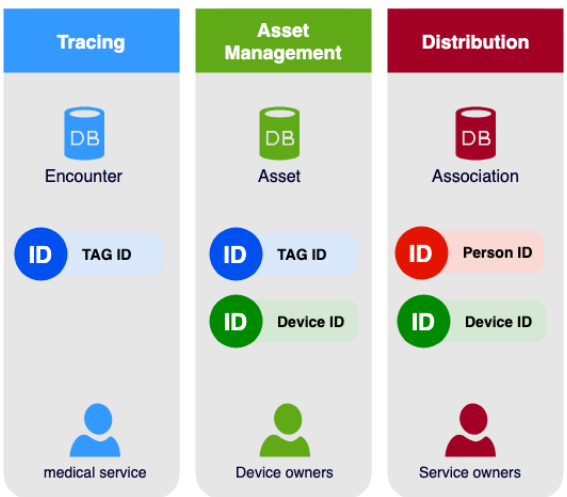


Figure 4: Global architecture based on scattered databases.

Three distinct databases were designed and deployed for that purpose: (1) Encounter, (2) Asset and (3) Association. Figure 4 shows the databases, the team having access and their main function. Access to the databases is only granted to specific teams with a clear purpose: e.g. medical service only has access to the Encounter DB, device owners only has access to the Asset DB and the registration service only has access to the Association DB.

An access to an individual database is not enough to relate the identity of a person and his/hers assigned proximeter guaranteeing anonymity. Also it shows the main identifiers (IDs) allowing a data pseudonymization which permits the data records be less identifiable while remaining suitable for the traceability (Table 3).

Table 3: Data Pseudonymization with Identifiers (IDs)

ID	Function
Person ID	CERN person identification number
Device ID	Proximeter visible identification number
TAG ID	Proximeter internal identification number

The CERN office of data protection (ODP) played a central role in the validation of the choices made. It particularly advised about the final solution providing users with the ability to consult their personal data kept at all times. To comply with this requirement the software suite added a new functionality called *self-service* to allow people to see their encounters. Obviously the data provided to the users listed the encounters of the person without the identification of the peers to ensure their anonymity.

Another aspect which was also looked at carefully was the security. The software suite together with the infrastructure was provided to the IT security team who run a complete audit to identify the robustness of the solution with respect to any possible data leak.

THE PROXIMETER PORTAL

The design of the software application was driven by the requirements of the contact tracing but also with the idea of minimising resources by employing services already provided by CERN. The data scattered in different databases imposed several constraints on the design of the software increasing the complexity and making difficult the diagnostics in case of any technical issue. A second concern was to make a software suite in the shortest possible period of time to provide the application to the medical service as soon as possible.

The software suite ensures functionality to manage the proximeters by the registration service: i.e. assignment, lost&found and relinquish (REGISTRATION, STATUS and FOUND), the self-service (MY DATA) allowing all users to consult the personal data retained (encounters) and the tracing (TRACKING) of close contacts for the use of the medical service (Fig. 5).

The portal application is based on web technologies and split in a front-end and a back-end. The web application back-end is based on *Java™ 11*, *Maven*™ and *Spring®*

[1] The Processing of Personal Data at CERN (OC11)

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV010

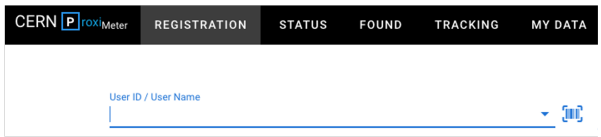


Figure 5: Proximeter portal main interface.

Boot and exposes data via a REST APIs and serves the static files of the front-end, built with *Vue.js*.

The code is hosted on CERN's *GitLab* and the build of the project, the publishing of the artifacts and the actual deploy of the application is managed through *GitLab's* pipelines. The final artifact resulting from the build process is a Docker image containing the layered *fat JAR* representing the application which is pushed to the project's *GitLab* Container Registry. The image is then deployed on CERN's PaaS (Platform as a Service - *OpenShift®*). The project is completed with metrics collection (*Prometheus*) and visualization (*Grafana*).

In order to provide the web application functionalities, many already existing CERN's resources has been leveraged. CERN's SSO (*Keycloak*) together with its integration with e-groups granted us the possibility to create group of users and give them different roles.

THE STATISTICS

The proximeter portal has been used since early 2021. The distribution of the proximeters reached a notable number of 5000 right before being mandatory at CERN and passed 8000 in September 2021. The only available figure of their real use is the number of those which produce encounters, otherwise the number of proximeters in use is unknown as we intentionally do not check on the proximeters when they connect to the network at their arrival at CERN.

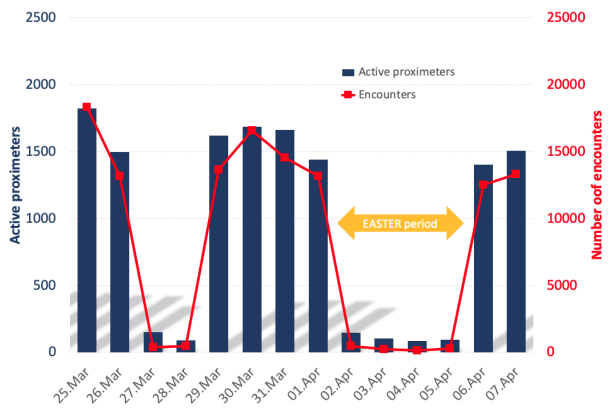


Figure 6: Active proximeters and number of encounters (per day).

During the initial period of the proximeter use (March 2021), the number of unique active proximeters was about 2000 and the number of encounters per working day was approximately of 14500 as average (Fig. 6), note the Easter holidays and week-ends with far fewer encounters. The number of close contacts (encounters with more than 15 minutes) was estimated of approximately 550 per day in average. Note the period shown had a high rate of personnel working from home (teleworking) so a limited number of CERN personnel working on site.

During 2021 the number of positive cases was up to 17 per week (5 per week in average), but the medical service checked on possible close contacts of personnel with compatible disease symptoms. As a reference, in the last weeks of September, the medical service executed approximately 10 extractions per week to find close contacts.

CONCLUSION

CERN invested in performing a rigorous contact tracing as a way to break the chain of infection. The key factor relied on improving the manual tracing executed by the medical service and provided an **immediate report** of close contacts for the medical service. This would minimise the number of employees who would need to be quarantined. The proximeter service revealed itself essential to allow a more precise contact tracing.

The success of this project, allowed by a notable internal cooperation among diverse teams at CERN, was not only based on the device itself but also on providing a service permitting the medical service to fulfil its needs of tracing. The service came to operation early 2021 and more than 8000 devices were distributed to CERN members of personnel who received the device as a protection measure while respecting the personnel privacy.

The software suite handling the proximeters and enabling the contact tracing was designed and engineered reusing all possible resources already available, therefore minimising cost and impact of CERN infrastructure and resources and, always, with the main concern of data privacy. This project indeed reflects well the concept of **privacy by design**.

REFERENCES

[1] WHO, "Contact tracing in the context of COVID-19," in *COVID-19: Surveillance, case investigation and epidemiological protocols*, 2021, https://www.who.int/publications/i/item/contact-tracing-in-the-context-of-covid-19

[2] Terabee, https://www.terabee.com

[3] Terabee Proximeter, https://www.terabee.com/shop/covid-products/terabee-proximeter/

[4] C. Merscher, R. Sierra, A. Zimmaro, M. Giordano, and S. Danzeca, "Proximeter CERN detecting device for personnel," in *25th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2021)*, 2021, doi:10.1051/epjconf/202125104025

[5] LoRaWAN® Specification, https://lora-alliance.org/about-lorawan/

[6] R. Sierra and H. Odziemczyk, "Readying CERN for connected device era," in *24th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2019)*, 2020, doi:10.1051/epjconf/202024507015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

THE INCLUSION OF WHITE RABBIT INTO THE GLOBAL INDUSTRY STANDARD IEEE 1588

M. Lipiński*, CERN, 1211 Geneva 23, Switzerland

Abstract

White Rabbit (WR) is the first CERN-born technology that has been incorporated into a global industry standard governed by the Institute of Electrical and Electronics Engineers (IEEE), the IEEE 1588 Precision Time Protocol (PTP). This showcase of technology transfer has been beneficial to both the standard and to WR technology. For the standard, it has allowed the PTP synchronisation performance to be increased by several orders of magnitude, opening new markets and opportunities for PTP implementers. While for WR technology, the review during its standardisation and its adoption by industry makes it future-proof and drives down prices of the WR hardware that is widely used in scientific installations. This article provides an insight into the 7-year-long WR standardisation process, describing its motivation, benefits, costs and the final result. After a short introduction to WR, it describes the process of reviewing, generalising and translating it into an IEEE standard. Finally, it retrospectively evaluates this process in terms of efforts and benefits to conclude that basing new technologies on standards and extending them bears short-term costs that bring long-term benefits.

INTRODUCTION

White Rabbit

White Rabbit (WR, [1]) is an open-source [2] technology developed to provide the Large Hadron Collider (LHC) accelerator chain with deterministic data transfer, synchronisation with sub-nanosecond accuracy and a precision of a few picoseconds. First used in 2012, the technology has since then expanded its applications [3] outside the field of particle physics and is now deployed in numerous scientific infrastructures worldwide. Moreover, it has shown its innovative potential by being commercialised and introduced into different industries, including telecommunications, financial markets, smart grids, the space industry and quantum computing. On 16 June 2020, the WR synchronisation method was recognised by being included in the worldwide industry IEEE 1588 standard called Precision Time Protocol (PTP, [4]), governed by the Institute of Electrical and Electronics Engineers (IEEE, [5]), the world's largest technical professional organisation dedicated to advancing technology.

WR's wide-spread use in scientific installations and its adoption into industry can be attributed to 3 factors:

1. The open-source and commercial nature.
2. The collaborative and welcoming community.
3. Basing it on standards, extending them if needed.

This article focuses on the third factor to the success of WR.

Standards and Standard-Defining Organisations

A technical standard is usually a formal document that establishes uniform engineering or technical criteria for externally visible operational aspects of a technology. While independently developed products (e.g. devices, software) that follow the same standard are meant to inter-operate, innovation is made possible in the internal implementation of these standard-based solutions.

Technical standards are prepared by groups that gather expert representatives of industry and academia specialised in a given domain. These groups are referred to as standard-defining organisations (SDOs) and can be organised as international organisations, unions, associations or consortia. Examples of SDOs relevant to this article:

1. International Telecommunication Union (ITU): a specialised agency of the United Nations responsible for information and communication technologies.
2. Institute of Electrical and Electronics Engineers (IEEE): a professional association for electronic engineering and electrical engineering (and associated disciplines).

Our everyday life is heavily dependent on standards prepared by IEEE and ITU. The ITU standards govern the operation of mobile networks. Commonly used means of communication, such as Ethernet, Local Area Networks and WiFi are all defined in IEEE standards.

IEEE1588 Standard

The IEEE 1588 standard defines the Precision Time Protocol (PTP) that provides precise synchronisation of clocks in packet-based networked systems. The standard is commonly used to synchronise devices (e.g. sensors, actuators) in factories, power plants, distributed measurement systems as well as in finance, audio-video and telecommunication. First published in 2002, it was updated in 2008 and 2019.

The operation of PTP is divided into two main actions: establishment of a synchronisation hierarchy in the network and synchronisation of network devices following the established hierarchy. The IEEE 1588 standard is actually a "generic framework" that defines the above-mentioned mechanisms, as well as numerous options and parameters to enhance and fine tune its operation. The configuration of this "generic framework" is called a "PTP Profile". The framework cannot be used without a profile, thus the IEEE 1588 standard includes generic-purpose profiles, called Default PTP Profiles. Many industries define PTP Profiles adjusted to their particular requirements [6]. For example ITU-T defines PTP Telecom Profiles (e.g. G.8265.1) and the IEEE defines PTP Profiles for Power (C37.238) as well as Audio/Video, Automation and Automotive (802.1AS).

Notably, the IEEE 1588 standard purposely does not define requirements with respect to synchronisation perfor-

* maciej.lipinski@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

mance or hardware. These requirements can be defined in the PTP Profiles. As such, synchronisation performance of PTP devices depends on their implementation and the PTP Profile used. Typically PTP devices provide synchronisation at the microsecond level.

WR AND ITS MOTIVATION FOR USING AND EXTENDING STANDARDS

The WR project was launched to renovate CERN's General Machine Timing (GMT), a system that orchestrates and synchronises operation of all the CERN accelerators. The GMT was created over two decades ago as a custom-made system based on the RS-422 standard. As such, it requires costly in-house support and tailor-made spare parts, resulting in maintenance challenges.

Apart from the technical requirements [7], the goal of the WR project was to create a technology with decades-long support from industry to minimise the cost of spare parts and maintenance. Achieving this goal was based on 3 pillars which are complementary: (1) open-source and commercial nature of WR, (2) collaborative and welcoming WR community, and (3) choice of well-established standards as the basis of WR. The public availability of design sources allows enhancement and reuse, as well as commercialisation by multiple companies. The more applications and users, the more attractive commercialisation is for companies while preventing vendor lock-in. The more companies sell WR devices, the more competitive is the price of WR gear, encouraging new users. Both companies and users favour investing in standard-based solutions for a number of reasons, such as stability and maturity, availability of existing products and tools, and interoperability of products from different vendors.

As even standard technologies become obsolete (e.g. RS-422) the choice of the base standard(s) is important. WR is based on the family of well-established IEEE 802 networking standards (IEEE 802.1Q and IEEE 802.3) and IEEE 1588, with good provisions for evolution and long-term support.

While the chosen base standards provided the functionalities (data transfer, synchronisation) required from WR, CERN's requirements exceeded their performance capabilities. At that time it was decided that any enhancements to improve the performance should be made compatible with the base standards for the following reasons:

1. Off-the-shelf standard gear can be used to minimise costs where high performance is not needed.
2. Off-the-shelf standard existing tools can be used.
3. The improvements are more likely to be implemented by companies.
4. The improvements can evolve with the standard allowing long-term support.
5. The improvements might make it into the base standard.

The above reasoning shaped the way in which the WR synchronisation method was specified, i.e. a WR extension to the IEEE 1588 standard.

WR EXTENSION TO IEEE 1588

CERN's requirement for the WR technology specified synchronisation at the sub-nanosecond level, which was well beyond the performance achievable by IEEE 1588 devices at the time of WR's conception. Within the WR project, the standard's shortcomings were identified and solutions proposed to overcome them:

1. Clock syntonisation over the physical layer.
2. Enhancement of timestamps through phase detection.
3. Automatic evaluation and calibration of asymmetries.

While initially, it was considered to specify the above PTP enhancements as an independent protocol around the operation of the Default PTP Profile, the flexibility of PTP's "generic framework" allowed to use its built-in extensions mechanisms to accommodate the proposed solutions. And thus, the WR synchronisation method is described in the *White Rabbit Specification - draft for comments* [8] as an extension of the IEEE 1588-2008 standard in the form of a White Rabbit PTP Profile (WR PTP Profile). After substantial efforts and consultations with the author of IEEE 1588, this profile was defined such that it is compatible with the Default PTP Profile and it looks as any other PTP Profile. The WR PTP Profile specified in [8] includes:

1. Theoretical background for the extensions.
2. Hardware requirements to implement the extensions.
3. Protocol additions/changes to support the extensions.
4. Formal definition of the WR PTP Profile.

Such a definition of the WR synchronisation method meant that in a distant future the WR PTP Profile could be standardised on its own or it could be incorporated into the base standard - an ambitious task that CERN decided to undertake.

PATHS FOR WR STANDARDISATION

There is no crash-course at CERN on how to standardise new technologies. All had to be learned from scratch on a trial and error basis. The WR standardisation effort was started by setting up a project [9] with the goal *"of standardising the White Rabbit extension to the IEEE1588-2008 standard"* and gathering a Study Group [10] to advise on how to do so.

First, potential WR standardisation paths were identified:

1. Standardisation in ITU-T:
   (a) within an existing Telecom PTP Profile,
   (b) as a new Telecom PTP Profile.
2. Standardisation in IEEE:
   (a) as a new PTP Profile included in IEEE 1588,
   (b) as part of AVB Gen2[1] in the 802.1AS PTP Profile.
3. Standardisation by a WR consortium acting as an SDO.

Second, each of the paths was studied in terms of cost, effort and prospects of success. As such, WR was presented at relevant ITU-T and IEEE meetings to the respective communities. Attendance in these meetings allowed to evaluate the operational procedures and dynamics of the groups, as

[1] Now known as Time Sensitive Networks.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

well as the level of interest in the extensions proposed by WR, and synergies with ongoing projects. The observations and conclusions from this phase were as follows:

1. The level of performance provided by WR exceeded the needs of industry at that time, thus companies would find it hard to justify efforts into WR standardisation.
2. The ITU-T path was considered infeasible.
3. Creating a WR Consortium was too costly and risky.
4. Many synergies were found between WR solutions and the AVB Gen2, which was encouraging.
5. WR inclusion into IEEE1588 was identified as the best and most feasible WR standardisation path.

At that time, WR was presented at important events such as the IEEE Plenary [11], as well as meetings of dedicated SDO working groups. The presentations were made by CERN and industry representatives and demonstrated applications outside of CERN, thanks to the open-source and collaborative nature of WR and its community. As a result of these efforts, WR became recognised in the industrial timing community. This, in turn, made it possible to consider WR for inclusion into the base standard - the best possible scenario.

WR STANDARDISATION IN IEEE 1588

IEEE Standard Lifecycle

Each IEEE standard must obey the *Standards Development Lifecycle* presented in Fig. 1 which for existing standards ensures that a new revision is published every 10 years. Luckily, step 1 in the lifecycle of IEEE 1588 was triggered in 2012 when WR standardisation paths were evaluated.

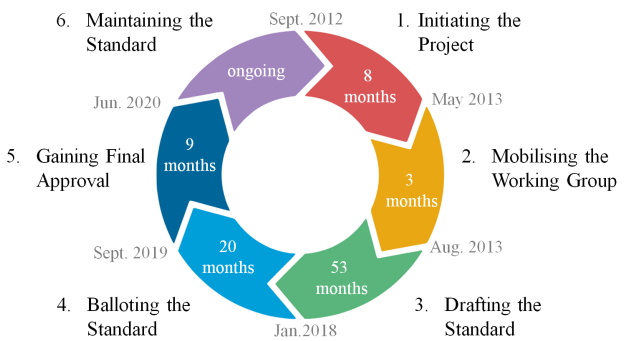


Figure 1: Standards Development Lifecycle, indicating times for the IEEE1588-2019 revision.

In the first step in Fig. 1, the scope of the project is defined in the Project Authorisation Request (PAR) which is submitted to IEEE for approval. For existing standards, the scope can be limited to a simple erratum or it can add new features. For the IEEE 1588, a Special Session at the ISPCS 2012[2] conference was held to evaluate the needs of the community. Among many contributions, it was proposed to include WR into the revised IEEE 1588 [12]. The feedback indicated that a major revision of the standard was needed, therefore a Study Group was formed to prepare the PAR which stated: *"The protocol enhances support for synchronization to better than 1 nanosecond"*. This sentence reflected group's enthusiasm to include WR into the standard, yet with no guarantees.

P1588 Working Group

In the second step in Fig. 1, a P1588 Working Group (P1588 WG, [13]) was mobilised to work on the new revision of IEEE 1588. Over 180 representatives from industry and academia jointed the group, though only around 40 actively participated. The P1588 WG was divided into 5 subcommittees (SC) to work on different aspects of the revision, namely Maintenance, Management, Architecture, Security and High Accuracy SCs were formed. The latter, was dedicated to including WR into the standard. Each SC met online twice a month and reported its progress monthly on an online Plenary of the entire P1588 WG. Two to three times a year, P1588 WG face-to-face (F2F) meetings were organised in different locations; one was held at CERN. The proposals prepared by SCs were reviewed by the entire P1588 WG.

P1588 High Accuracy Subcommittee

The High Accuracy SC was lead by CERN and included around 20 active representatives from such institutes and companies as NIST, Huawei, National Instruments, ADVA Optical Networking, Microsemi, Intel, Boeing, Ericsson, Meinberg and Cisco. The SC started step 3 in Fig. 1 by studying and understanding the WR PTP Profile [8]. This was important before deciding how to include WR into the standard. The study revealed a number of challenges:

1. There was no immediate need for the performance provided by WR which required hardware modifications. The interest was rather in using WR methods for improving performance of existing hardware.
2. WR is specified for a particular type of medium (1 Gbps Ethernet over single mode bidirectional fibre) while the standard is generic.
3. WR requires a particular hardware implementation while IEEE 1588 is purposely silent in these regards.

As a result, the WR PTP Profile could not be included into the IEEE 1588 standard in a straightforward way and the efforts for WR inclusion were questioned numerous times.

Eventually, a consensus was reached which proposed a method of including WR in the standard that encouraged contributions from the SC members. As described in Fig. 2:

1. WR was divided into a number of optional features that could be useful on their own for different industries, i.e. Layer 1 Syntonisation Enhancements (L1 Sync), Configurable correction of timestamps, Calculation of the delayAsymmetry, mechanism for external configuration and masterOnly mode.
2. These optional features were made generic and extended to accommodate the needs of some members [14]. Methods known to work for WR became a particle configuration of the generic feature and were presented as examples.

[2] The International IEEE Symposium on Precision Clock Synchronization for Measurement, Control, & Communication.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

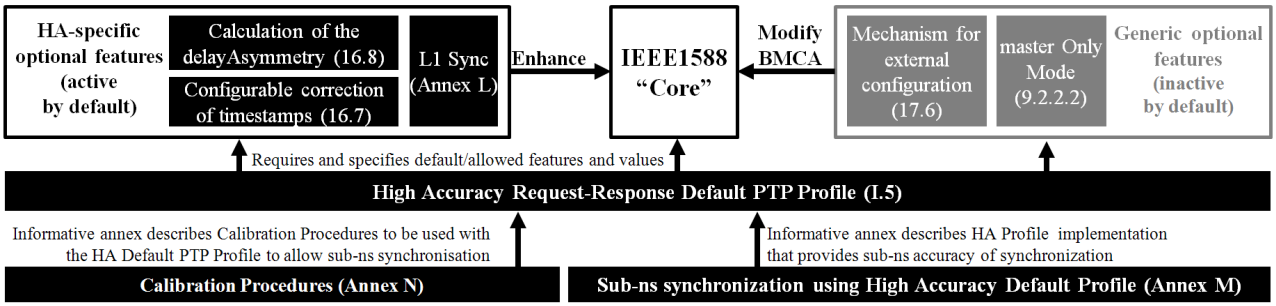


Figure 2: WR incorporation into IEEE1588-2019.

3. A new High-Accuracy Default PTP Profile (HA PTP Profile) was defined that makes use of the new optional features and specifies how a WR implementation should configure them.
4. While the WR methods included into the normative parts of the standard define only protocol aspects (no requirements on performance or hardware), the implementation details that allow achieving sub-ns accuracy of synchronisation were described in an informative annex, as an example implementation of the HA PTP Profile. Additionally, the WR calibration procedures [15] could be also included as an informative annex.

With the above compromise methodology, all aspects of the WR PTP Profile [8] and WR Calibration [15] could be included into the IEEE 1588 standard. WR was translated into three new annexes (L, M and N) and a number of additional clauses (e.g. 16.7, 16.8, 17.6, I.5) constituting a total of over 55 pages. Additionally, the core text of the standard (PTP "generic framework") had to be enhanced with more precise definitions (e.g. clock, clock signal, Direct PTP Link), as well as an improved specification of timestamp generation and medium asymmetry, see [16] for details.

This compromise means that the WR PTP Profile and the HA PTP Profile are not compatible at the protocol level. While the existing WR hardware can be used to implement the new profile, the software WR implementation of the PTP stack [17] had to be updated. Before submitting the final text for the HA features and PTP Profile to the P1588 WG, the updated protocol was prototyped and compliance tests were developed independently to ensure its high quality [18].

With the new text ready and the solutions thoroughly tested, the High Accuracy SC submitted the text to the P1588 WG, were it joined that of the other SCs - the draft of the entire new standard (which grew from 289 to 499 pages) was ready for balloting.

Balloting

Balloting is a voting and review process of IEEE standards in which each reviewer votes to either reject or approve the text. The approval can be made conditional on implementing voter's comments. Each comment must be precise, explain the problem, and suggest new text that solves it. All comments are reviewed by a dedicated committee and each comment must be addressed. An updated text of the standard is submitted for re-circulation. In re-circulation, the reviewers vote with the same rules as before, but comments can be made only to the text that changed. In this way, after a limited number of re-circulations, the voting converges.

For the revised version of IEEE 1588, there were three ballots internal to the P1588 WG, which are still step 3 in Fig. 1. In these WG ballots, 3352 comments were submitted in the first one, 1504 in the second one and 749 in the third one. It took 18 months to go through this step and have the draft ready for the next step, i.e. Sponsor Ballot.

The Sponsor Ballot, step 4 in Fig. 1, is the IEEE Standards Association Public Review process in which anyone in the world can take part and vote submitting comments. A group of 127 registered balloters submitted a total of 358 comments which took 2 re-circulations and a total of 16 months to resolve before getting the required number of affirmative votes.

Final Approval and Publication

The balloted draft was then approved by the IEEE Standards Review Committee and Standards Board in November 2019. At this point the new revision of the IEEE 1588 standard, IEEE1588-2019, superseded the previous IEEE1588-2008. However, the standard was not yet publicly available. It was passed to the IEEE Editor. The editing process took around 6 months and the new edition of the IEEE 1588 standard, WR included, was published on 16 June 2020.

COSTS, BENEFITS AND LESSONS LEARNED

On behalf of CERN, the author was heavily involved in each step of the IEEE *Standards Development Lifecycle* described in the previous section. His involvement included leading the High Accuracy SC, contributing to the work of the Architecture, Management and Maintenance SCs, as well as substantially helping in the ballot comments resolution and working with the IEEE Editor in the final stage.

The indicative costs of CERN's involvement in the work of the P1588 WG in order to standardise WR amounts to an estimated 2.25 person-years and 26 kCHF spread over 7 years. It includes participation in online and F2F meetings, preparation and review of documents and organisation of a F2F meeting at CERN; it excludes implementation of the HA PTP Profile and its compliance tests.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The motivation for standardising WR was to achieve long-term benefits, such as wide adoption and support from industry to minimise costs for CERN installations. Although these benefits cannot yet be unequivocally claimed only 1 year after IEEE 1588-2019 publication, some clear initial indications can be observed. Moreover other benefits of WR standardisation are apparent.

WR technology has benefited technically from the transformation into the HA options and PTP Profile. Not only has a number of known shortcomings of WR been fixed but WR technology has also become a more generic and flexible solution – see [14].

Even before the WR standardisation was completed, many companies had enthusiastically acted on the premise that WR would become an integral part of IEEE 1588. This included companies already familiar with WR and newcomers. The prospective WR standardisation clearly motivated these companies to invest time and resources into WR technology, which was tested for future applications in telecommunication networks on this premise [19]. The increasing number of applications and companies producing and supporting WR devices is essential for healthy competition and technology evolution, e.g. commercial long-distance WR-based time transfer [20].

The IEEE 1588 standard benefited from CERN's contributions and expertise. The constantly growing number of WR users increases the popularity of the standard. The enhanced level of accuracy achievable by the IEEE 1588-2019 meets the increasing needs of the industry. While the accuracy required from IEEE 1588 devices by industry in 2012 was at sub-µs level, it has now increased to 4 ns [21] and will surely increase further. IEEE 1588-2019 is prepared to meet these requirements.

Finally, WR standardisation is a technology transfer that fulfils CERN's mission to *push the frontiers of science and technology, for the benefit of all* [22]. CERN has become a recognised player in the timing industry and continues its mission as the key driving force for innovation in the P1588 WG that continues to oversee the evolution of the IEEE1588 standard. As such CERN now leads the *P1588 New Features SC* mandated to add new features to IEEE1588, some of them inspired by new developments in the WR community.

WR AFTER STANDARDISATION

WR technology provides deterministic data transfer and accurate time transfer over an Ethernet Bridged Local Area Network using an extension to IEEE 1588. Initially this extension was the WR PTP Profile of IEEE1588-2008. After the WR standardisation described in this article, it will be the HA PTP Profile of IEEE1588-2019. This update requires only upgrade of the software implementation of the PTP used in WR devices, i.e. PPSi [17]. Currently the beta version of the HA PTP Profile is available, yet official releases of WR firmware still use the WR PTP Profile.

The migration of WR devices to use the HA PTP Profile is foreseen in the coming years and it is intended to be transparent for the WR users. WR Switches will support both PTP Profiles, WR and HA. The WR Nodes will use a profile of their choice, yet their support for the WR PTP Profile is meant to be deprecated at some point. On the other hand, the WR Switch is meant to support both profiles in the long-term. When connecting a WR device (switch or node) to a WR Switch, it will attempt to run the HA PTP Profile by default. If the HA PTP Profile is not detected on the adjacent WR device, the WR PTP Profile will be attempted. Thus, a WR Network can effectively use both profiles, WR and HA, in a manner transparent to the user.

The compliance of WR devices with the WR PTP Profile and the HA PTP Profile is ensured through extensive compliance tests developed using the Veryx ATTEST framework. While the framework is proprietary, the compliance tests are open-source [18]. The WR PTP Profile tests were purposely developed to ensure compliance of new WR firmware releases with the legacy devices/release. The HA PTP Profile tests were developed to verify the new profile during its development, and are meant to verify compliance with the HA PTP Profile of its implementations by CERN and the industry.

Notably, WR technology is a particular implementation of the HA PTP Profile that guarantees sub-ns accuracy of synchronisation. There might be other implementations of that profile without such guarantee.

CONCLUSION

Standardisation of a new technology is a long process which requires not only an excellent technical solution but also patience and an open-minded attitude which allows to reach consensus. The duration and dynamics of the standardisation process depend on the SDO, the particular standard and the group that works on it, thus each potential standardisation effort must be considered individually.

The author is convinced that for technologies which are meant to last for decades and be deployed in thousands of units, the benefits of standardisation clearly out-weigh the efforts. A decrease of price by a hypothetical 50 CHF for a single WR device, costing hundreds or thousands CHF, due to competition encouraged by the WR standardisation compensates the monetary cost of this standardisation as soon as 500 unites are purchased. The 2.25 person-years spent during 7 years on WR standardisation are more than compensated by 20 experts from industry scrutinising the technology ensuring its longevity and making it future-proof, while learning how to implement it in their companies.

Of course, standardisation is an investment which, by nature, brings profits with time and it is only in a decade or so that a true evaluation of the benefits of WR standardisation will be possible.

ACKNOWLEDGEMENTS

WR would not have made it into the IEEE 1588 standard without the strong support and invaluable help of John Eidson, called in the community "the father" of IEEE1588.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV011

REFERENCES

[1] White Rabbit Project – official CERN website, https://www.cern.ch/white-rabbit

[2] White Rabbit Project Open Hardware Repository, https://ohwr.org/project/white-rabbit/wikis

[3] M. Lipiński *et al.*, "White Rabbit Applications and Enhancements", in *Proc. ISPCS2018*, CERN, Geneva, Switzerland, Oct. 2018, pp. 106-112. https://ohwr.org/project/white-rabbit/uploads/7f9e67258850d5c036629a509bf2e124/ISPCS2018-WRApplicatoinsAndEnhancements.pdf

[4] *IEEE 1588-2019 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard, New York, NY, USA, Nov. 2019; https://standards.ieee.org/standard/1588-2019.html

[5] Institute of Electrical and Electronics Engineers (IEEE), https://www.ieee.org/

[6] Precision Time Protocol Profiles, https://sagroups.ieee.org/1588/ptp-profiles

[7] M. Lipiński, T. Włostowski, J. Serrano, and P. Alvarez, "White Rabbit: a PTP application for robust sub-nanosecond synchronization", in *Proc.* ISPCS2011, Munich, Germany, Sep. 2011, pp. 25-30. doi:10.1109/ISPCS.2011.6070148

[8] E. Cota, M. Lipiński, T. Włostowski, E. van der Bij and J. Serrano, "White Rabbit Specification: Draft for Comments, version 2.0", CERN, Geneva, Switzerland, Jul. 2011; https://white-rabbit.web.cern.ch/documents/WhiteRabbitSpec.v2.0.pdf

[9] White Rabbit Standardization Project, https://ohwr.org/project/wr-std/wikis/home

[10] White Rabbit Study Group, https://ohwr.org/project/wr-std/wikis/std-group

[11] M. Lipiński, "White Rabbit - Ethernet-based solution for sub-ns synchronization and deterministic, reliable data delivery", *Tutorials*, IEEE 802 Plenary, Geneva, Switzerland, Jul. 2013. https://www.ieee802.org/802_tutorials/2013-07/WR_Tutorial_IEEE.pdf

[12] M. Lipiński, "High accuracy extension/option/profile", in *Special Session* ISPCS2012, San Francisco, USA, Sep. 2012. https://www.ohwr.org/project/white-rabbit/uploads/b0c0f513bf8734bc856f6860fa693201/ISPCS2012-specialSession-final.pdf

[13] P1588 Working Group – official website, https://sagroups.ieee.org/1588

[14] O. Ronen and M. Lipiński, "Enhanced Synchronization Accuracy in IEEE1588", in *Proc.* ISPCS2015, Beijing, China, Oct. 2015, pp. 76-81. doi:10.1109/ISPCS.2015.7324687

[15] G. Daniluk, *White Rabbit calibration procedure, version 1.1*, CERN, Geneva, Switzerland, Nov 2015; https://white-rabbit.web.cern.ch/documents/WR_Calibration-v1.1-20151109.pdf

[16] WR integration into IEEE1588-2019 as High Accuracy, https://ohwr.org/projects/wr-std/wiki/wrin1588

[17] PPSi: PTP Ported to Silicon, https://ohwr.org/project/ppsi/wikis/home

[18] White Rabbit Compliance Tests, https://ohwr.org/project/wr-compliance-tests/wikis

[19] H. Imlau, *Highly Accurate Time Dissemination and Network Synchronization at ISPCS 2019*, Telekom Technik GmbH, *keynote*, ISPCS20219, Portland USA, Sep. 2019, https://www.researchgate.net/publication/336013265_Highly_Accurate_Time_Dissemination_and_Network_Synchronization_at_ISPCS_2019

[20] T. Cozzens, "White Rabbit makes leap for time over fiber", *GPS World*, Online, Sep. 2021. https://www.gpsworld.com/white-rabbit-makes-leap-for-time-over-fiber

[21] *ITU-T G.8272.1 - Timing characteristics of enhanced primary reference time clocks*, ITU-T Recommendation, Nov. 2016. doi:11.1002/1000/13162

[22] WHAT IS CERN'S MISSION? https://home.cern/about/who-we-are/our-mission

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

THE ESRF-EBS SIMULATOR: A COMMISSIONING BOOSTER

S. Liuzzo*, L.R. Carver, J.M. Chaize, L .Farvacque, A. Götz, D. Lacoste,
N .Leclercq, F. Poncet, E.Taurel†, S. White
ESRF, Grenoble, France

Abstract

The ESRF-Extremely Brilliant Source (ESRF-EBS) [1] is the first-of-a-kind fourth-generation high-energy synchrotron. After only a 20-month shutdown, scientific users were back to carry out experiments with the new source. The EBS Simulator (EBSS) played a major role in the success of the commissioning of the new storage ring. Acting as a development, sandbox and training platform, the machine simulator allowed control room applications and tools to be up and ready from day one. The EBSS can also be seen as the initial block of a storage ring digital twin. The present article provides an overview of the current status of the EBS Simulator and presents the current roadmap foreseen for its future.

INTRODUCTION

The ESRF storage ring was upgraded in 2019 to provide 100 times brigther X-rays [1]. The strong demand for experiments at ESRF imposed a very tight schedule for dismantling, installation and commissioning of the new storage ring. Overall, 20 months of dark time for external users were necessary [2]. Only three of these months were dedicated to Storage Ring commissioning. In order to cope with this schedule, the design and update of the whole software infrastructure had to be brought forward as much as possible. This is particularly true for the high-level applications needed on the first day of commissioning, such as the new magnets control. In addition, several applications specific to the commissioning such as beam threading algorithms [3] had to be prepared and tested to be used effectively, with minimal debug time.

For this purpose a full test of the software from high-level applications to power-supplies level (current input, not hardware level) was realized via an EBS control system simulator (EBSS). This control system simulator was strongly focused on the new EBS magnets control system that needed to be completely redesigned, but included as well all the frequently used diagnostic devices (beam position monitor (BPMs), tunes, emittances, etc..) needed for the development of the tools used for the commissioning. The output values of the simulated diagnostic devices are generated from a given lattice optics model that is updated upon a magnetic strength or RF frequency variations in the simulated control system - as depicted in Fig. 1.

From the user's point of view, the simulator is identical to the "physical" control system. The devices providing the information on the beam position have identical names for

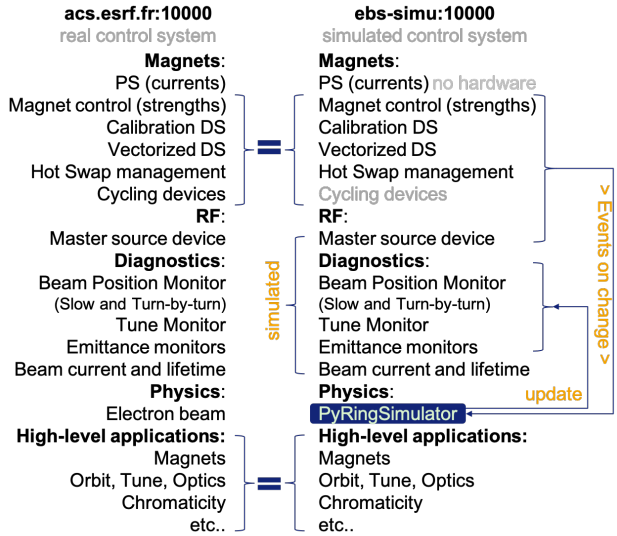


Figure 1: Real control system vs EBS simulator.

attributes and properties as their physical counterparts. This allows to port the applications from the simulator environment to the real machine by simply changing the environment variable pointing to the TANGO [4] database.

EBSS is a clone of a subset of the EBS control system that allows to interact with a simulated beam and to visualize the expected behaviour of most of the relevant electron beam and lattice observables: orbit, tunes, emittances, optics and coupling.

In the past, the ability to run off-line control room applications was already available, for example via toolkits such as the Matlab-Middle-Layer [5], used in many 3rd generation light sources or specific solutions - like the one on which the Virtual XFEL [6] relies.

For storage-rings, Matlab-Middle-Layer provides a high level switch from simulations to real beam experiments. This limits the development to anything above the matlab-middle-layer software infrastructure. The EBSS instead acts at a lower level, replacing directly the beam, thus giving access to all control levels. It notably enables real-time tests of features (1-2s/loop) not only for beam dynamics experts but also for control system engineers.

STRUCTURE OF THE SIMULATOR

The core of an EBSS instance [7] is composed of more than ~ 4000 Tango devices compared to ~ 25000 in the physical accelerator complex (including the 3 accelerators - linac, booster + storage ring). Each EBSS instance runs on its own Tango control system - the associated Tango database

* simone.liuzzo@esrf.fr
† taurel@esrf.fr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

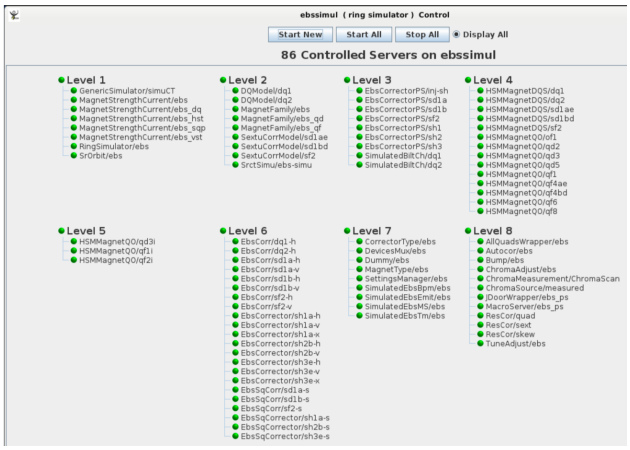


Figure 2: View of the 86 Tango device servers running within a docker.

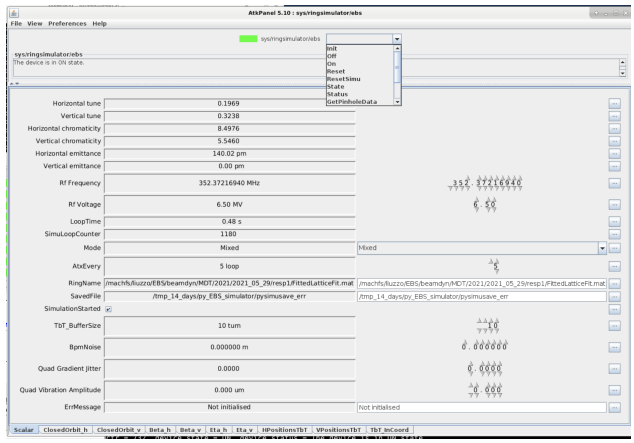


Figure 3: PyRingSimulator ATK panel view.

being isolated in a Docker container. This allows multiple EBSS to run simultaneously and independently on the same host. The device servers architecture is depicted in Fig. 2. Presently three EBSS instances are deployed at the ESRF - each one consuming about 25-30% of the CPU resources and 18-20% of the memory of a 16-CPUs/64-Gb host.

Tango Device Servers

Each EBSS Tango device exposes the exact same interface as its production counterpart. This ensures a total transparency to its clients and guarantees a smooth transition from simulation to production. At the lowest level of the control system, most of the devices are written in C++ and support a compilation flag providing a way to specify the target platform - i.e., simulation or production. For instance, the BPMs read the beam position from the dedicated hardware when running in production mode, whereas they obtain it from the ring simulator while running in the context of an EBSS instance. The behaviour of some devices has also been modified in order to broadcast or specifically notify (upon change) the EBSS environment - e.g., a strength modification on a magnet will trigger a modification of the EBS storage ring magnetic model and a consequent update of the relevant beam parameters. These asynchronous notifications rely only on the Tango events mechanism. A total of ~ 100 Device Classes and ~ 500 Device Servers are configured to run the EBSS.

Physics Core: PyRingSimulator The PyRingSimulator device server (see Fig. 3, and Fig. 2 level 1) runs high energy electron beam dynamics simulations [8]. It was initially developed in Matlab [9] and C++ [10], and has recently been ported to Python 3 [11].

The PyRingSimulator DS runs a dedicated process at startup (`ebs_simu.py`) that continuously[1] computes the optics using python Accelerator Toolbox (pyAT) [12]. For each EBSS instance, one CPU core is then fully dedicated to this process. This structure allows to better exploit the CPU resources and provides a clear separation among the issues and workload related to control and those related purely to beam dynamics. Moreover, this separation allows for future extensions of the beam dynamic simulations to exploit more CPU resources, GPUs or the ESRF computing cluster, keeping the pyTANGO DS unchanged.

The simulated diagnostic DS continuously reads/monitors the updated parameters in PyRingSimulator instead of measuring real beam properties. From the user's point of view, the devices providing orbit, tune, chromaticity, etc, behave as the physical ones available in the control room. Figure 1 depicts the structure of the Tango devices layers: the PyRingSimulator device is the only one not existing in the real control system: it actually replaces the electron beam.

The electron beam magnetic lattice (optics model) used by PyRingSimulator can be the same as the one used for operation, a measured one or a different one (yet, an EBS optics model). It is defined by three individual lattice files: a reference optics with no errors nor corrections, a lattice including errors (on any magnet or BPM, potentially unstable) and a lattice including the same errors as in the previous one and the corresponding corrections. This structure allows to hide the lattice errors (any error that is possible to specify with pyAT) to the user, just as in real life. On the other hand, these three models can be identical if needed to give full freedom in the conditions of the machine to mimic. For example, to simulate the first turns steering, the lattice with errors and corrections has all the corrections set to zero, and it is thus identical to the one with errors only. An unstable lattice is simulated and as the correctors strengths are set to progress in the first turns steering, the simulations are updated to display the trajectory signal at more and more BPMs, finally leading to a stable machine (see [3]).

In order to make the EBSS even closer to the physical storage ring a few stochastic effects are also included: BPM noise, magnets vibrations (quadrupole), and field gradient jitter.

The optics computations can be selected (for speed reasons) among the following options:

[1] the loop runs continuously and not only upon changes to allow the introduction of stochastic errors, see later.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

1. *Emittance* mode (>~ 1.6 s/loop), computing all linear optics, including emittances but excluding Turn-by-turn data;
2. *LinearOptics* mode, as the *Emittance* mode but without the lengthy natural emittance computation (> 0.8s/loop),
3. *Mixed*, iterates one loop *Emittance* and few loops *LinearOptics*,
4. *Turn-by-turn* mode (0.02 s/loop/turn (not linear, 0.4 s for 1 turn, 2 s for 100 turns), only computes the T-b-T buffer for the required number of turns and with the specified injection coordinates.

The actual loop speed of the EBSS depends on the number of magnets (or RF) that are modified before the required computations. The times are approximately doubled when using radiation for all computations (rarely used case). A computation loop counter is used at the Tango device level as a trigger signal for new data.

The PyRingSimulator has the ability to set the strengths of the magnets from the optics model (with errors and corrections) to the magnets of the EBSS. This feature was introduced to be able to reset the initial conditions of the simulations and proved to be extremely useful.

Moreover it is possible to save a lattice to pyAT format at any moment for further studies or to debug the EBSS itself.

Simulated Lifetime/Current Decay The lifetime/current device is a standalone device implementing an exponential decay starting from a given current, with a given lifetime. Simulations of beam (Touschek) lifetime are impractically long (hours on a single core, minutes on a computing cluster), and are not crucial for the purpose of the EBSS.

DS AND APPLICATIONS DEVELOPED FOR EBSS

Before the commissioning of the EBS storage ring, the EBSS proved to be extremely useful. Several high level applications needed complete refurbishment, including some major ones such as the magnet control applications.

Magnets Control

Having to develop a completely new control of the magnets - starting from low level device servers to a user-friendly and intuitive graphical interface - was the occasion to also include new features. This is a risky action when it is not possible to test the new features. For EBS, the EBSS was the key to progress in this development, allowing to debug offline in a pure virtual/simulated environment. The new features included in the magnet control were:

- individual control of (more than 1000) power-supplies,
- control of magnets organized by arrays or family,
- magnet set points expressed in “strength” (calibration curves integrated in the control system, including combined function sextupoles with 5 current channels to pilot 4 strengths),

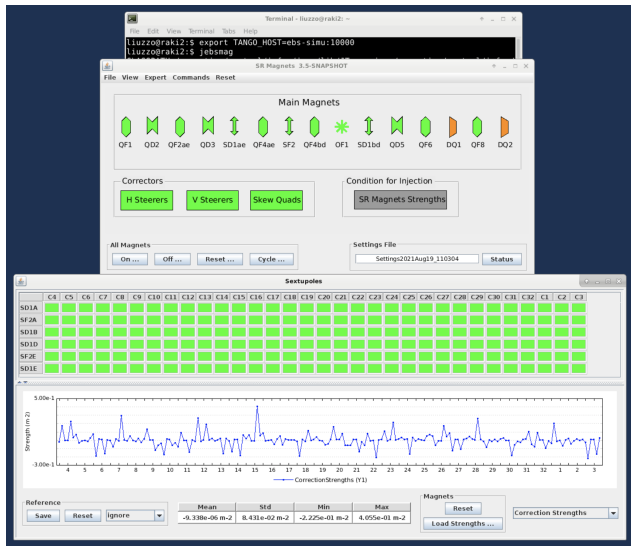


Figure 4: Application for the control of the individual magnets in strengths. Top, the main panel with family tuning, bottom, a view of the sextupole magnets.

- definition of new cycling and ON sequences that avoid the peak power consumption to run out of limits,
- ability to *freeze* magnets (inhibit any user modification of a specific magnet),
- integrate the hot-swap features [13]
- tuning of electron beam resonance knobs,
- save and load of currents (primary) and strengths (utility) of the storage ring magnets configuration files.
- a completely new user interface.

All these features were introduced and tested within the EBSS, leaving for the commissioning only a few issues and some unforeseen developments. A-posteriori, even more applications of the EBSS would have been beneficial before the commissioning - for example, to identify calibration issues.

The final look and feel of the EBS magnets control application is shown in Fig. 4. From the user’s point of view, it is impossible to distinguish an instance of the application running in the context of the simulator from an other attached to the physical machine. All actions are fully functional.

First Turns Steering

An other key feature to a successful commissioning was the ability to automate the first turns trajectory steering. For this purpose basic scripts were designed and tested in the EBSS again solving bugs, giving confidence in the use of the tools and leaving only a few developments to be tested with real beam. The successful use of the EBSS was here also enhanced by the test of applications designed for EBS but tested in the old ESRF storage ring. More details are available in Ref. [3].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

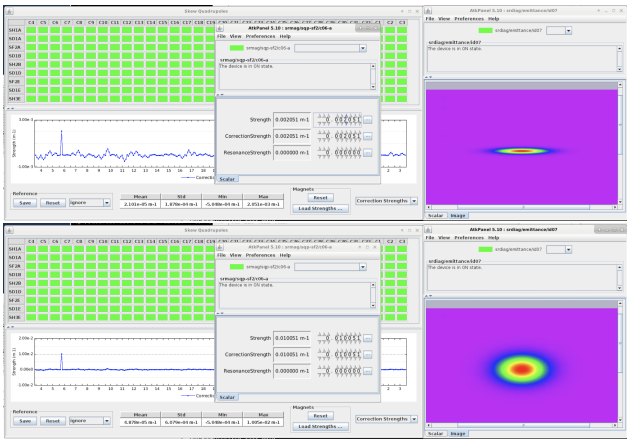


Figure 5: One skew quadrupole is powered (top, low field, bottom, large field) in the EBSS to show the effect on one of the simulated emittance diagnostics.

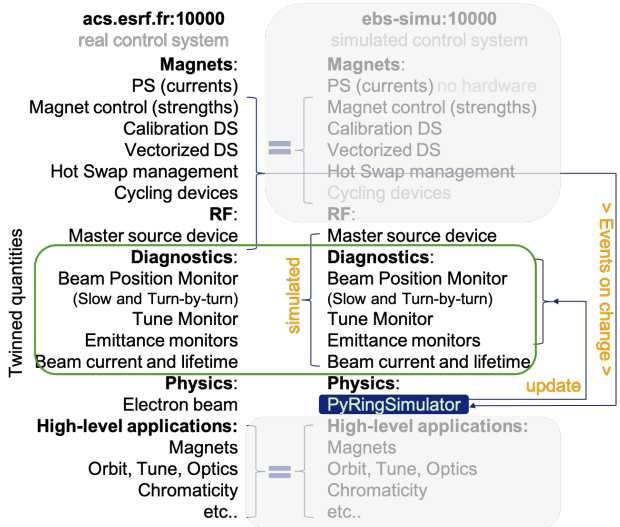


Figure 6: Digital twin configuration using the existing EBS Simulator.

Tune Correction

A new tune correction application was also deployed. The multiplication of the available knobs allowed to generalize the tune correction to any vector of quadrupole magnets, rather than simply two families. This new feature was tested in the EBSS and in the old ESRF storage ring and did not require any further tuning with real beam.

Optics and Coupling Correction

Some of the applications used during the operation of the previous ESRF storage ring could not be updated before commissioning because of the lack of time. Some were deliberately postponed with no detrimental impact on the commissioning schedule. An example is the optics and coupling correction application. This was completely redesigned and updated during the beamline commissioning time, making extensive use of the EBSS simulator to test all steps, from the response matrix measurement to the fit of the lattice errors and the application of quadrupole correction. An example of beam emittance display with a skew quadrupole mistuned on purpose is displayed in Fig. 5.

Slow and Fast Orbit Correction Interaction

The EBSS simulator has nevertheless a limitation, that is the speed of the computation loop. All fast systems such as fast orbit and emittance feedbacks are not available in the EBSS. Nevertheless, tests of the correct simultaneous use of Fast Orbit Feedback, Slow orbit correction, and setting of closed orbit bumps, were possible and allowed to exploit the EBSS to redesign the orbit correction logic for EBS in parallel to the User Service Mode operation (USM).

More Applications Developed for Commissioning

The EBS simulator also allowed the development and test of: beam based alignment functions, chromaticity measurement and correction, a Tango DS for the computation of optics at any given longitudinal coordinate, a procedure of measurement to evaluate the beam energy, the procedure to load new SR optics, and many more.

EBSS allows to perform realistic, real-time virtual Machine Dedicated Time (MDT). During real MDTs if a problem occurs, the solution can be looked for in parallel with the EBSS, while other MDT activities continue to take place.

TOWARDS A DIGITAL TWIN

The EBSS infrastructure is a first step towards a Digital Twin [14] of the Storage Ring. In fact, a complete digital twin continuously monitors the real machine settings and updates the expected output accordingly. Today it is manually possible to use a measured lattice to trigger the simulator computations and to set the strengths so that EBSS matches the real machine configuration. In order to move towards a digital-twin, this action would need to be automated. An instance of the PyRingSimulator device could then run in the real control system sourcing the currently loaded optics as a model - a trivial action requiring a simple modification of a Tango property - and use the real magnets settings - that are already expressed in strengths! - to update the optics model (see Fig. 6). This would allow to have constantly (at the speed of the simulator loop) an estimate of the expected orbit, tunes, dispersion and optics of the SR. The absolute comparison being far from easy to obtain with the existing lattice model at the ESRF, relative variations monitoring should be possible without major issues. A further improvement would be to include ID gaps models, such that it would be possible to monitor for example the optics modulations induced by ID gaps during USM.

Both improvements on the lattice modelling and on the control infrastructure are needed, since the magnet device servers are presently designed to work either on the real control system or in the EBSS. Regarding the control aspects of this problem, the idea would be to implement a bridge between the physical control system and its virtual counter-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

part. The ability to access Tango devices across separated control systems makes such a task quite straightforward.

Another step towards a real digital twin is to connect the control device to either the real live devices or the archive database to display live or historical values as readings. This step would be useful for debugging applications, new algorithms and or using the digital twin as a time machine able to go back in time. Connecting the digital twin to the beamline control system simulator is another area to be explored. This would allow beamlines to test developments with a simulated copy of the accelerator i.e. the EBSS.

PORTING EBSS

There are a number of new projects started or planned to replace the current 3rd generation storage rings with a 4th generation one. For those projects which are using Tango for the accelerator controls or are interested in trying it out with a simulator, the EBSS offers a powerful solution. However there is a certain amount of work required to port the EBSS to a new storage ring.

To port the EBSS to another storage ring the following would need to be carried out:

- Define the devices according to the naming scheme of the new storage ring
- Implement simulation devices for each of the device classes. One option is to start off with a generic device simulator e.g. using the tango-simlib [15]
- Populate the Tango database with the simulated devices
- Define the optics of the new storage ring for pyAT
- Start the docker containers with the new simulators

CONCLUSIONS

The EBSS proved to be an extremely powerful tool for the preparation of the ESRF-EBS storage ring commissioning. It allowed to eradicate most of the coarse bugs and to include several additional features without any delay on the initial schedule. The EBSS was used with large profit also after the EBS storage ring commissioning and is presently used for the continuous development and upgrade of all the softwares for which a direct impact on the electron beam is expected.

The infrastructure that was set in place to realize the EBSS is such that future developments towards a Digital Twin are simplified. The final objective will be to continuously monitor the expected (simulated) beam properties from the storage ring control room, such as optics and response to magnetic fields variations. Such a development will also allow to implement Artificial Intelligence and Machine Learning algorithm that will constantly compare model vs measurement enabling: 1) early detection of faults, 2) slow drifts of parameters potentially linked to hardware failures, 3) continuous lattice optics model update, and 4) many other features.

The extension to other accelerators is presently not trivial, due to the intrinsic peculiarity of each control system. Nevertheless the strategy is set and at least the PyRingSimulator DS would require few modifications (the specific name of the devices) to be used for other storage rings, or high energy electron transfer lines. Specific developments will be needed instead to extend the use of the simulator to linacs, energy ramped accelerators and hadron accelerators.

The several aspects mentioned above will be addressed in the near future and are presently part of a proposal for a future EU collaboration project [16].

REFERENCES

[1] J. Biasci *et al.*, "A low-emittance lattice for the esrf," *Synchrotron Radiation News*, vol. 27, no. 6, pp. 8–12, 2014. DOI: 10.1080/08940886.2014.970931.

[2] S. White *et al.*, "Commissioning and Restart of ESRF-EBS," in *Proc. IPAC'21*, (Campinas, SP, Brazil), ser. International Particle Accelerator Conference, JACoW Publishing, Geneva, Switzerland, Aug. 2021, MOXA01, pp. 1–6, ISBN: 978-3-95450-214-1. DOI: 10.18429/JACoW-IPAC2021-MOXA01.

[3] S. Liuzzo *et al.*, "Preparation of the ebs beam commissioning," *Journal of Physics: Conference Series*, vol. 1350, p. 012 022, Nov. 2019. DOI: 10.1088/1742-6596/1350/1/012022.

[4] A. Götz *et al.*, "State of the Tango Controls Kernel Development in 2019," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1234–1239, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-WEPHA058.

[5] G. Portmann, J. Corbett, and A. Terebilo, "An accelerator control middle layer using matlab," in *Proceedings of the 2005 Particle Accelerator Conference*, 2005, pp. 4009–4011. DOI: 10.1109/PAC.2005.1591699.

[6] R. Kammering *et al.*, "The Virtual European XFEL Accelerator," in *Proc. of International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, 17-23 October 2015*, (Melbourne, Australia), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, Geneva, Switzerland: JACoW, Dec. 2015, pp. 578–580, ISBN: 978-3-95450-148-9. DOI: doi:10.18429/JACoW-ICALEPCS2015-TUD3O04.

[7] E. Taurel, D. Lacoste, and N. Leclercq. "Ebs simulator tango device servers," https://gitlab.esrf.fr/accelerators/Simulators/EbsSimulator (accessed: 08.10.2021).

[8] S. Liuzzo. "Python ring simulator optics computation," https://gitlab.esrf.fr/BeamDynamics/pythontools/optics_for_pyringsimulator (accessed: 08.10.2021).

[9] *Matlab*, The MathWorks, Natick, MA, USA, <2020a>.

[10] B. Stroustrup, "Thriving in a crowded and changing world: C++ 2006-2020," *Proc. ACM Program. Lang.*, vol. 4, no. HOPL, 70:1–70:168, 2020. DOI: 10.1145/3386320.

[11] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[12] W. Rogers, N. Carmignani, L. Farvacque, and B. Nash, "pyAT: A Python Build of Accelerator Toolbox," in *8th International Particle Accelerator Conference*, Copenhagen, Denmark, May 2017, THPAB060. DOI: 10.18429/JACoW-IPAC2017-THPAB060.

[13] ESRF, internal communication, 2012–2021.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV012

[14] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020, ISSN: 2169-3536. DOI: 10.1109/access.2020.2998358.

[15] T. community. "Tango-simlib: Easily generate tango device simulators," https://github.com/ska-sa/tango-simlib (accessed: 05.10.2021).

[16] D. T. P. for Analytical Research Infrastructure Experiments (DiTARI), grant application proposal for European Commission's Horizon Europe framework programme HORIZON-INFRA-2021-TECH-01-01, 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV013

A DYNAMIC BEAM SCHEDULING SYSTEM FOR THE FAIR ACCELERATOR FACILITY

S. Krepp*, J. Fitzek, H. Hüther, R. Mueller, A. Schaller, A. Walter
GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany

Abstract

The new Accelerator Control System for GSI/FAIR is now being used productively for the GSI accelerator facility. As the central component for online beam orchestration, the Beam Scheduling System (BSS) is situated between the FAIR Settings Management System and the FAIR Timing System. Besides device settings, the Settings Management System provides timing schedules for beam production. The primary purpose of BSS is to define which of the beam schedules are executed by the Timing System, how often and in which order. To provide runtime decisions in pre-planned execution options (e.g. skipping of a particular beam), it processes external signals like user input, experiment requests or beam prohibits provided by the interlock system. More recently, advanced features have been added that allow for dynamic execution control required by Storage Ring Mode features such as breakpoints, repetitions, skipping and manipulations. This contribution gives an overview of the Beam Scheduling System including its interfaces.

INTRODUCTION

One of the major building blocks of the new Accelerator Control System for GSI/FAIR is the Beam Scheduling System (BSS). Residing in the middle tier of the FAIR Control System, its core functionality is the orchestration of beams based on user requests. Figure 1 shows how BSS is embedded into the overall Control System architecture.

In the FAIR Settings Management System LSA, beams are represented as Beam Production Chains and are put together to Patterns for defining an execution sequence. This includes both, settings for hardware devices, as well as an execution schedule defining which Timing Events are to be sent to which parts of the facility. Additionally, by assigning Patterns to Pattern Groups, LSA defines which Patterns must be executed sequentially and which Patterns may run in parallel. The schedule for each Pattern and the information about Pattern Groups is provided to the BSS system, which creates an overall schedule for the accelerator facility and sends it to the Timing System's Generator component. The Generator then translates this schedule to the low-level programming of the Data Master.

Operators and experimenters define, which beams they would like to have produced using applications and services, that in turn send these requests to BSS. At the same time, the Master Accelerator Status Processor (MASP) determines whether a certain Pattern can be executed, by collecting status and interlocks of all required devices and services. BSS then combines both of these inputs and sends commands to

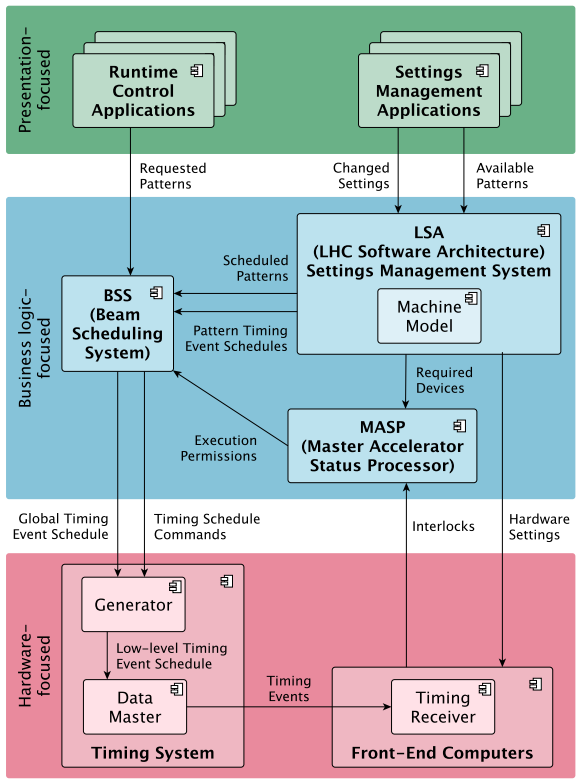


Figure 1: BSS in the FAIR Control System.

the Generator, that dynamically modifies the overall schedule accordingly. At runtime, the Data Master sends out timing events via the White Rabbit-based timing network. Those events lead to synchronous execution of settings in devices as calculated and provided by LSA.

As shown, BSS is the central instance that decides which of the preconfigured schedules are executed. This contribution presents technical details of the BSS system and its interfaces to give an insight, of how beam orchestration is realized within the GSI/FAIR Accelerator Control System.

SCHEDULE REPRESENTATION

Schedules within BSS are represented as schedule graphs that can be executed by the Timing System [1]. The graph itself is represented in the .dot format [2]. The vertices in this directed graph can roughly be interpreted as executable units linked together by a set of edges. Once started, a control thread walks over those vertices one after another in order to execute them. The actual sequence of vertices is defined by a set of edges marked as default destinations. Alternative routes through the schedule are defined by a set of alternative destinations.

* s.krepp@gsi.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV013

To allow for dynamic schedule changes, the Timing System supports switching of default destinations in the schedule graph, thus allowing the beam schedule to be altered at runtime by changing the default path through the schedule graph. BSS is responsible for managing the configuration of the Timing System by providing beam schedules and for altering those schedules on the fly based on user requests.

SCHEDULE CONFIGURATION

As described, LSA supplies BSS with individual schedules per Pattern and it is up to BSS to integrate them into a global timing schedule. Doing so, the schedules are not just merged, but extended to later gain flexibility at runtime. Conceptionally, the resulting schedule is driven by three major requirements. Firstly, Patterns specified in one Pattern Group must be scheduled sequentially respecting their specified execution order, while Patterns in different Pattern Groups shall run in parallel. Secondly, BSS must be able to permanently enable and disable each individual Pattern schedule's execution to allow for beam prohibits and user requests. Thirdly, BSS must be able to schedule individual Patterns for being executed exactly once.

The first requirement is met, as BSS manages Pattern Schedules on a per Pattern Group basis. It combines the individual Pattern schedules to a circular Pattern Group schedule graph. Different Pattern Groups result in different Pattern Group schedule graphs and can be executed in parallel.

An example Pattern Group schedule graph is shown in Fig. 2. Each individual Pattern schedule has one explicit entry and exit node, e.g. A_ENTRY and A_EXIT. The Pattern schedule for Pattern A is shown here only as one node A for the sake of simplicity. However, in the real schedule graph, node A actually consists of many vertices and edges that represent the actual behaviour of the machine, e.g. the timing events for this Pattern. The currently active loop through the graph is drawn with solid edges, alternative paths trough the graph are drawn with dotted edges.

To meet the second requirement, and to be able to switch the execution of one Pattern on and off, BSS adds an alternative path directly from ENTRY to EXIT, i.e. the edge from A_ENTRY to A_EXIT. To prevent Patterns from being executed right away, once they are provided to the Timing System's Generator and execution is started, the edge that skips Pattern execution is in fact set as the default one. Consequently, it is drawn as a solid edge in the figure. The edge used for actually executing Pattern A is represented by a dotted line, as it must be explicitly switched on later by BSS.

In addition to the Pattern schedules provided by LSA, BSS adds a so-called Default Pattern schedule to the Pattern Group schedule graph. It is used for idle operation, i.e. when no beam is requested. Finally, each Pattern's exit node is extended with outgoing edges to each available Pattern's entry node which gives BSS the ability to build arbitrary loops of Pattern schedules.

To meet the third requirement of being able to execute a Pattern only once, it was necessary to keep the entry nodes

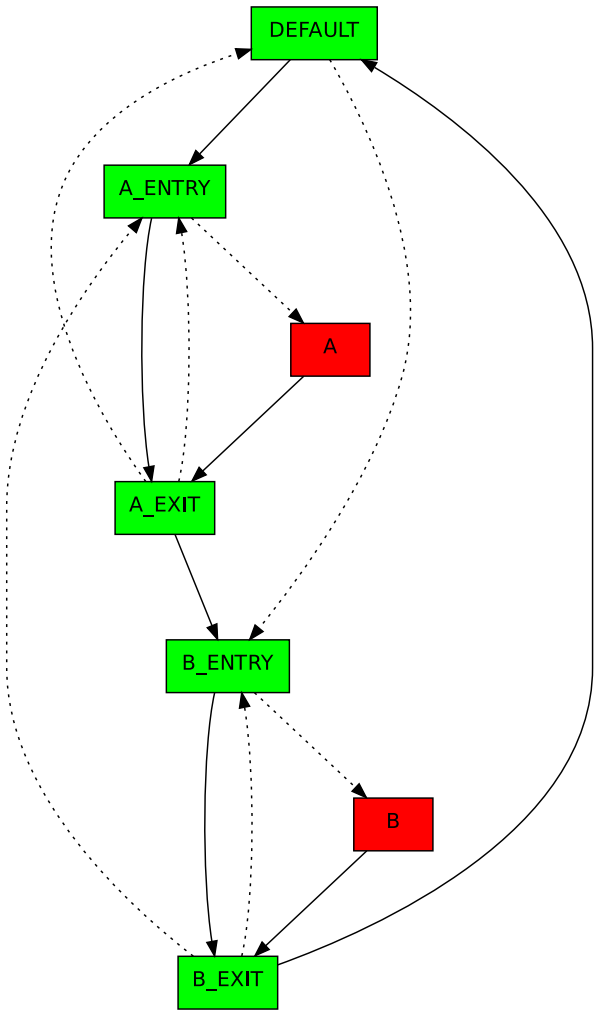


Figure 2: Pattern Group schedule with only the Default Pattern being active.

in the active loop for each Pattern also in the case, that this Pattern is not being requested at the moment. This is why in case no Pattern is requested, the Pattern Group schedule graph's active loop does not only consist of the Default Pattern pointing to itself, but contains all the entry and exit nodes as shown. The single shot request is described in more detail in section NON-PERMANENT OR SINGLE SHOT REQUESTS.

After the Pattern Group schedule has been prepared by BSS, it is added to the Timing System's global schedule. The Timing System supports multithreading by assigning parts of the global timing schedule to a certain thread. In order to support parallel execution of independent Patterns, each Pattern Group schedule is executed in its own thread. With a start command sent to the Generator, the execution begins right away. When the graph from Fig. 2 is executed, the Timing System loops over the nodes that are coloured green: a sequence containing the Default Pattern schedule and the nodes A_ENTRY, A_EXIT, B_ENTRY, B_EXIT. However, the Patterns A and B are not executed since their internal schedule subgraphs A and B are skipped.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Whenever the Pattern Group gets changed, e.g. a Pattern is added or removed, BSS stops the corresponding thread and replaces the old Pattern Group schedule with the new one.

As shown, the overall timing schedule at FAIR includes schedules for multiple Pattern Groups, which itself integrate multiple Pattern schedules. The following sections describe how BSS interacts with the Timing Schedule and how this is triggered by external requests.

SIGNAL-BASED RUNTIME DECISIONS

The primary input for runtime decisions concerning which parts of the global schedule should be executed are dedicated binary signals, with valid values being either enabled or disabled. Signal states are exclusively managed by BSS, however it exposes a client API for reading and writing signal states. BSS defines various signal types which can be used for different use cases as well as for access control. Signals of type DISABLED_BY_LSA for instance, can only be set by the Settings Management System itself and not be overridden by client applications.

Basically, BSS performs two categories of runtime decisions. The first one are decisions on Pattern Group level, i.e. which Pattern should be running, which will be described in the next section. The second category are decisions on Pattern level where decisions on alternative paths within a Pattern schedule are made (see section STORAGE RING CONTROL).

PATTERN SCHEDULE EXECUTION CONDITIONS

In order to decide which Pattern schedules to execute, each schedule is bound to an execution condition. Each execution condition is expressed as a propositional formula using signals as statements and logical operators (AND, OR, NOT). Because of their binary nature, signals can be directly mapped to boolean variables. This way, complex execution conditions can be easily built as the following example illustrates:

$$(SIGNAL_A \vee SIGNAL_B) \wedge \neg SIGNAL_C \qquad (1)$$

If a Pattern schedule is bound to the above condition, it is scheduled for execution if at least one of SIGNAL_A or SIGNAL_B is enabled and Signal_C is disabled.

Signals used in execution conditions are either predefined signals that are independent from current scheduling-related settings in the facility (static signals) or signals defined by the Settings Management System (dynamic signals). Static signals for instance, are typically used for requesting beam, also from custom experiment control systems. It is up to the Settings Management System to supply BSS with execution conditions and dynamic signal definitions.

PERMANENT REQUESTS

With signals being used in execution conditions and BSS providing a simple API for changing signal states, users can enable and disable Pattern executions by simply sending permanent signal change requests. Signal state changes are persisted in a database. Afterwards BSS executes relevant execution conditions and determines which Patterns to enable or disable in the global timing schedule. Finally, it sends commands to the Timing System that switch schedule edges accordingly.

In order to permanently enable an actual Pattern for execution, the default destination from the Pattern's entry node is simply switched from the exit note to its actual timing schedule content (as represented by A in Fig. 3).

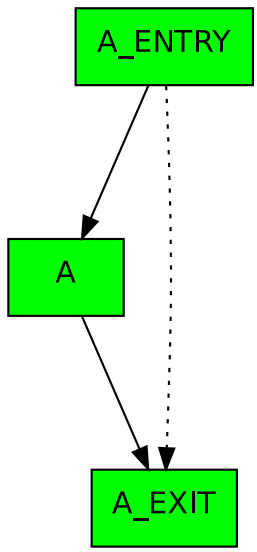


Figure 3: Part of a Pattern Group schedule with Pattern A being enabled.

NON-PERMANENT OR SINGLE SHOT REQUESTS

As described above, permanent signal requests lead to schedules being executed periodically since the resulting default execution path is always a circular subgraph. Therefore, those requests are not adequate for experiments requiring precisely a single shot or any other fixed number of Pattern executions. This is especially true for schedules of short duration, as reaction times of operators (and even automated systems) are threshold-bound and not arbitrarily short. It cannot be expected that an operator waits for a single execution of a Pattern and is able then to disable it before the next execution starts.

To allow for single shots, BSS provides so called non-permanent requests. Whenever a non-permanent request arrives, BSS does not apply the requested signal changes permanently, but builds a signal state snapshot from all existing signal states and applies the non-permanent changes to it. Afterwards, execution conditions of all schedules are evaluated against these snapshots. Schedules being enabled

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV013

by this evaluation are scheduled to be executed exactly once (Fig. 4).

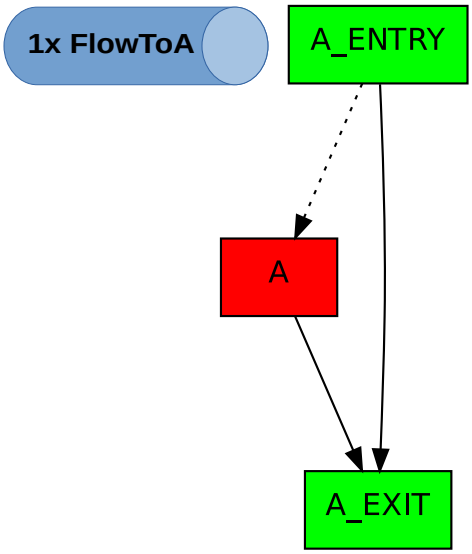


Figure 4: Queued Flow command in A_ENTRY leading to an exactly once execution of A when consumed.

To make this work, the BSS utilizes the ability of the Timing System to define command queues for schedule nodes. To prepare the schedule for non-permanent requests, a command queue is defined for each Pattern's entry node. Whenever a Pattern schedule should be executed exactly once, BSS writes a non-permanent flow command to the corresponding command queue specifying a temporary successor. When the Timing System executes this node, it will consume the command and will not go down the default execution path, but instead to the one specified in the command. Since the queued command is consumed, the prior default path is used again for the next execution.

STORAGE RING CONTROL

In addition to enabling and disabling whole Patterns, BSS also allows for a more fine-grained schedule control, that is extensively used for the Storage Ring Mode features breakpoints, skipping, repetition and manipulation for ESR and CRYRING [3]. To support these features, beam schedules for FAIR can contain sub-schedules with elements/edges that can be directly controlled by users utilizing BSS's signals. The mechanism is simple and is again based on switchable schedule edges which are based on the state of certain dynamic signals. A set of signal types has been added, that allows the definition of signals, from which BSS can directly infer which edges to switch.

To allow for breakpoints, where Pattern execution stops at defined points, the signal type BREAK_ENABLED is connected with a pre-defined break loop in the schedule graph. Setting the corresponding signal allows BSS to enable or disable the breakpoint. The name of the specific breakpoint is encoded in the signal's name, thus allowing for several breakpoints in the same schedule. Knowing the structure of

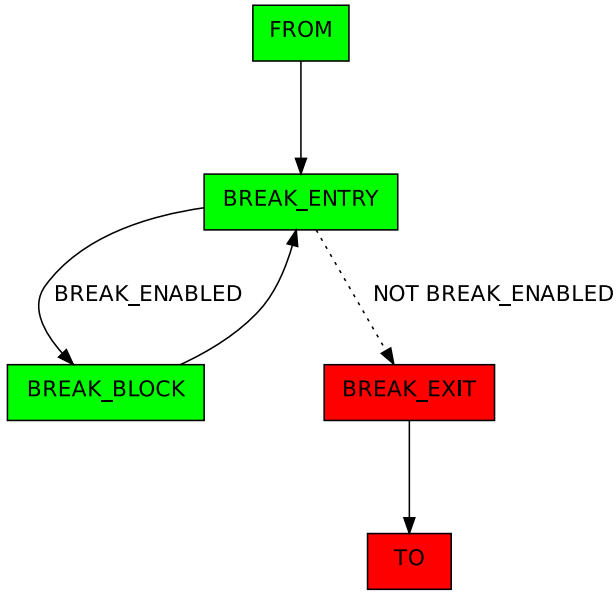


Figure 5: Part of Pattern schedule with an enabled breakpoint.

a breakpoint sub-graph, BSS can switch the edges accordingly. This is demonstrated in Fig. 5. If BSS receives a signal with type BREAK_ENABLED and value==ENABLED, BSS firstly decodes the breakpoint Subchain from the signal's name. Afterwards it sets the default execution path from BREAK_ENTRY to BREAK_BLOCK_ENTRY. If the same signal is received with value==DISABLED, it sets BREAK_EXIT as the successor of BREAK_ENTRY, which means that the break loop is no longer executed.

Skipping of Subchains has been realized using the signal type SKIP_SUBCHAIN. Similarly to skipping whole Patterns as described above, signals of this type are associated with edges that allow circumventing optional Subchains. As all of these features make use of dynamic signals whose initial value is set to be FALSE, the Subchain will be executed and not skipped by default.

Another basic feature of Storage Ring Mode is supporting Subchain repetitions, e.g. for performing a measurement for a defined number of times. With the beginning of Subchain execution in the Timing System, a command queue at an dedicated repetition node is filled with a command that defines the number of repetitions for this Subchain. BSS is responsible for cancelling these repetitions upon user request. This is done by flushing the corresponding queue using the Generator's command interface.

The most distinguished Storage Ring Mode feature is trimming (i.e. manipulation of settings) while a Pattern is running. This manipulation is realized as a loop in the graph that is connected with an exit signal that can be triggered by the user through BSS. While executing the same Subchain over and over again, the user can apply small setting changes to the devices.

For more information about these Storage Ring Mode features, please refer to [3].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

BEAM SCHEDULING VS. MACHINE PROTECTION

The Beam Scheduling System is not part of the Machine Protection System of the FAIR facility as it operates on a different time scale than fast hardware protection systems. However, if something goes wrong in the machine, BSS is responsible for preventing Pattern schedules from being executed on a software level. To do so, it continuously processes status messages from the Master Accelerator Status Processor (MASP), which in turn monitors relevant devices and services. Whenever BSS detects, that a Pattern must no longer be executed, BSS disables the associated schedule in the global timing graph. In contrast to Patterns that are just disabled by user requests, for prohibited Patterns, even their entry and exit nodes are excluded from the schedule loop (see Fig. 6). This ensures, that parts of the global schedule which have been prohibited by interlocks, can no longer be enabled by user requests, as long as the causing interlocks remain active. In this regard, interlocks act as an overruling of the signal based execution conditions.

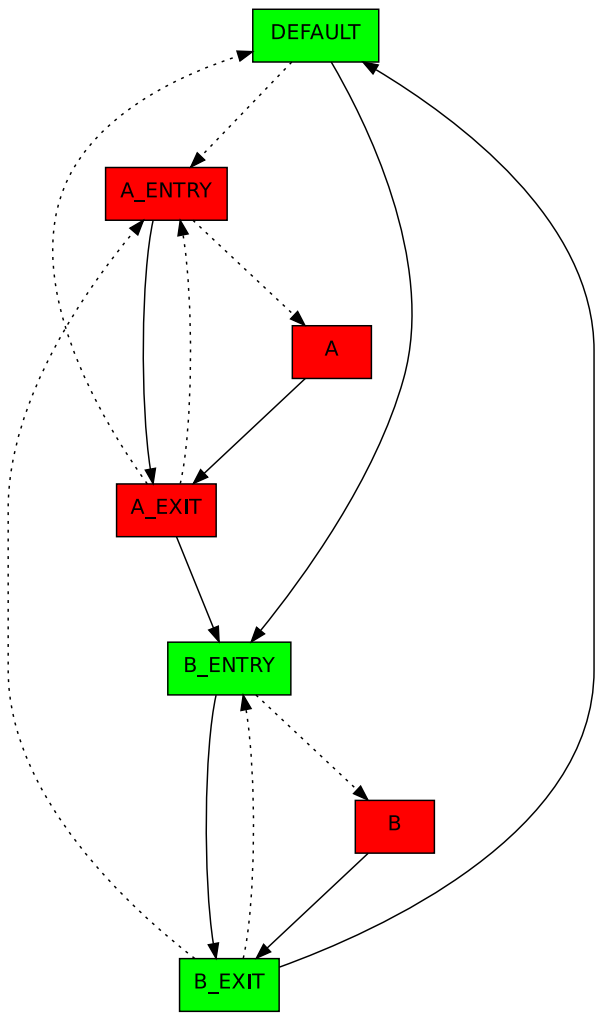


Figure 6: A Pattern Group schedule with Pattern A disabled by the Machine Protection System.

SUMMARY AND OUTLOOK

The Beam Scheduling System, as described in this contribution, has been used productively in several beamtimes. Newer features, like those for Storage Ring Mode operation have been added and it has proven to be a reliable and important building block of the FAIR Control System. Complete integration into the settings management system's data supply process, has allowed for an integrated solution, that makes LSA not only the place for calculating settings for devices, but also the place for schedule planning. BSS on the other hand, creates the global execution schedule and brings it together with runtime information, like user and experiment requests and beam prohibits, which makes it the central instance to decide, which of the preconfigured schedules are to be executed by the Timing System.

Next BSS developments will target requirements coming from the Injector Controls Upgrade project, which will bring the existing UNILAC into the new Accelerator Control System. Challenges are expected to arise from its 50Hz operation and a large number of pre-planned schedules.

REFERENCES

[1] DataMaster Manual, https://www-acc.gsi.de/wiki/pub/Timing/TimingSystemDataMaster/FTN_dm_schedules.pdf

[2] DOT Language, https://graphviz.org/doc/info/lang.html

[3] R. Mueller, J. Fitzek, H. C. Hüther, H. Liebermann, D. Ondreka, A. Schaller, and A. Walter, "Supporting Flexible Runtime Control and Storage Ring Operation with the FAIR Settings Management System", presented at 21th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper WEPV047, this conference.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV014

UPGRADE OF THE NewSUBARU CONTROL SYSTEM

N. Hosoda[†1,2], T. Fukui[2], Y. Hamada[1], M. Ishii[1,2], A. Kiyomichi[1], K. Okada[1], T. Sugimoto[1,2]
[1]Japan Synchrotron Radiation Research Institute, Hyogo, Japan
[2]RIKEN SPring-8 Center, Hyogo, Japan

Abstract

A new dedicated electron injector linear accelerator (linac) was constructed for the soft X-ray synchrotron radiation facility, NewSUBARU. The SPring-8 linac was used as the injector at the facility since its operation started. However, the new linac enabled NewSUBARU to operate independently from the SPring-8. The control system of the new linac and the existing storage ring must be constructed as a unified system for seamless operation. The control framework already used for the SACLA/SPring-8 was also used for the NewSUBARU. MicroTCA.4 (MTCA.4), EtherCAT, and GigE vision camera were used for the new linac control. For the storage ring control, the existing VMEbus system (VME) was used with virtually no changes. The open source version of Qt5 was selected to create a graphical user interface program (GUI). Additionally, the design of the new linac is planned to be used in the 3 GeV synchrotron radiation facility project currently under construction in eastern Japan. Similar kind of hardware and control framework will be utilized for the project.

INTRODUCTION

NewSUBARU is a soft X-ray synchrotron radiation facility with a 1.5 GeV electron storage ring having a circumference of 118 m [1]. It was built at the SPring-8 site and shared a linac of SPring-8 as an injector. The NewSUBARU started user experiments in January 2000. SPring-8 is a hard X-ray synchrotron radiation facility with an 8 GeV electron storage ring having a circumference of 1,436 m. It has a 140 m-long 1 GeV linac and an 8 GeV booster synchrotron having a circumference of 396 m as injectors and has been in service for user experiments since October 1997. SACLA is an X-ray free electron laser facility with a 400 m-long 8 GeV linac that was constructed at the SPring-8 site. The SACLA has been in service since March 2012.

In the SPring-8 upgrade project (SPring-8-II), a method for realizing an ultra-low emittance ring was studied. The plan consisted of drastically modifying the SPring-8 storage ring and injecting a low emittance beam of SACLA directly into the ring. This required the existing injectors, the linac, and the booster synchrotron of SPring-8, to be shut down. However, this would further show that NewSUBARU would no longer be operational. Therefore, a new 1 GeV linac had to be built exclusively for NewSUBARU. As it had to be installed in the existing beam transport tunnel, the linac was designed to be compact with a total length of 70 m and used C-band accelerating structures developed at the SACLA [2]. In July 2020, a beam injection from the SPring-8 linac to the NewSUBARU was completed and the installation of the new linac was started. The commissioning of the linac started in February 2021, and two months later, in April, the user experiments at NewSUBARU were resumed. Figure 1 shows the layout before and after the new linac was installed at the NewSUBARU. A picture showing the NewSUBARU building and the new annex building for the new linac is illustrated in Figure 2.

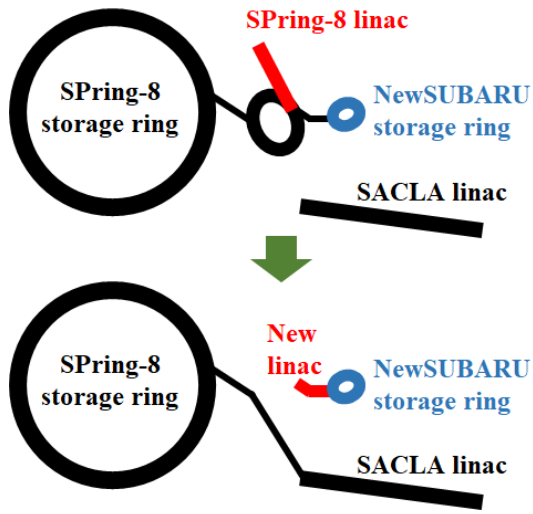


Figure 1: Layout of the NewSUBARU before and after installation of the new linac.

Figure 2: NewSUBARU building and new annex building for the new linac.

Since 1995, MADOCA control framework has been developed and used for the SPring-8 and the SACLA [3]. In 2018, the MADOCA was upgraded for integrated operation of the SACLA and SPring-8 as part of the SPring-8-II [4-6]. Using this new control system, beam injection from the SACLA to the SPring-8 storage ring was started in 2020.

The NewSUBARU has adopted MADOCA as its control system since its introduction. With the construction of the new linac, it was natural to completely shift from the first MADOCA to the upgraded one. To operate the new linac and the existing storage ring together, a file server, the database servers, and the computer networks required further upgrading.

When upgrading an existing facility, the shutdown period must be as brief as possible. To achieve this, advance

† hosoda@spring8.or.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

careful preparation and an efficient work plan for the shutdown period are vital. It was expected that changing the VME of the storage ring would require tremendous work, such as, checking the I/O of the equipment. Hence, it was decided to leave the VME mostly unchanged. In contrast, all the servers, networks, and software were upgraded.

This paper reports the upgraded NewSUBARU control system.

EQUIPMENT CONTROL OF THE NEW LINAC

MTCA.4

The RF system of the new linac consists of a gridded RF thermionic gun, a 238 MHz RF cavity, a 476 MHz subharmonic buncher, an S-band accelerating structure, and 16 high-gradient C-band accelerating structures. The low-level RF (LLRF) system of the SPring-8 storage ring was updated from conventional analog circuits to digital circuits using the MTCA.4 in 2016. From then on, the MTCA.4 has been operating for the LLRF system [7, 8]. Additionally, the MTCA.4 was selected for the LLRF system of the new linac.

As a high-speed AD/DA board for IQ modulation and demodulation of the RF signals, the Struck SIS8325 AMC board was utilized. This board had 10-channels of 250 Ms/s 16-bit ADCs, 2-channels 250 Ms/s 16-bit DACs, and an FPGA. For the FPGA logic, the following additional features were implemented along with the original logic: converting the data detected by the ADCs to the IQ signals; collecting the data at a fixed delay time from a master trigger signal of the linac; storing the waveforms in memory banks; and reading them out using a DMA transfer.

For the Rear Transition Modules (RTMs), which are responsible for the RF signal I/O and level adjustment, a newly manufactured Candox 72DSR238A01 direct sampling board was used for 238 MHz and Struck DWC8VM1 down-conversion boards were used for other frequencies.

In 2018, a test bench of a gridded RF thermionic gun, a 238 MHz RF cavity, and a 476 MHz sub-harmonic buncher for the linac was constructed and the operation of the MTCA.4 was tested for them.

The C-band accelerating structure system for the linac was tested in May 2020 to evaluate its performance, including the LLRF system using an MTCA.4. The device control software running on the MTCA.4 CPU was developed in conjunction with the test.

The same SIS8325 AMC boards were used for the readout of the nine beam position monitors (BPMs) and five beam current transformers (CTs).

A newly developed MDT MMETRG01B AMC board was used to transmit a trigger signal. This AMC board had five SFP transceivers (one input and four outputs) on the front panel and was connected to the other boards via optical fibers. A high-speed serial signal with a bit rate of 1 Gbps and 8B/10B encoding was used to transmit the event code that defines when the trigger signals should be output. The delay time of the trigger signal could be set using a 24-bit coarse delay counter with a 238 MHz clock and a 5-bit fine delay counter with 78 ps resolution. The board could handle 16 inputs and 16 outputs with a dedicated level converter connected via a connector mounted on the front panel. Furthermore, eight M-LVDS lines of the MTCA.4 backplane could be used for input and output, and the trigger signals were transmitted to the SIS8325s installed in the same chassis without any additional wiring.

The MMETRG01B was installed in the LLRF section of the storage ring to generate and transmit a master trigger signal. Four MMETRG01Bs were installed in the LLRF sections of the annex building for the new linac to receive the master trigger signal. They were connected using phase-stabilized optical fibers to avoid any temperature drift effects. Table 1 summarizes the AMC modules used in the five MTCA.4.

Table 1: AMC Modules Used in the Five MTCA.4

Host	Target equipment	AMC modules
1	linac master trigger, storage ring injection magnets/monitors trigger	MMETRG01B × 1
2	linac LLRF/HPRF/vacuum of gun, 238, 476	MMETRG01B × 1, AdXMC1573 × 1, SIS8325 x3
3	linac LLRF/HPRF/vacuum of S-band, BPM × 3, CT × 4	MMETRG01B × 1, AdXMC1573 × 1, SIS8325 × 3
4	linac LLRF/HPRF/vacuum of C-band1, C-band2, BPM × 2	MMETRG01B ×1, AdXMC1573 × 1, SIS8325 × 5
5	linac LLRF/HPRF/vacuum of C-band3, C-band4, BPM × 4, CT × 1	MMETRG01B × 1, AdXMC1573 × 1, SIS8325 × 6

The MicroTCA Carrier Hub (MCH) and CPU board used were from N.A.T., and the operating system (OS) was the Ubuntu 16.04 LTS low latency kernel. The MCH had an intelligent platform management interface (IPMI) to monitor the status of each AMC board, air-cooling fan, power supply, and so on, from other servers via a network. We are testing to monitor those monitor values from a Zabbix server.

EtherCAT

The MTCA.4 was used for the devices to treat high-speed wide-bandwidth data, whereas, the EtherCAT was used for relatively slow-speed devices [9].

PLCs were used in high-power RF (HPRF), such as, klystron power supplies and vacuum controllers. The PLCs were controlled using EtherCAT. The Advanet AdXMC1573 XMC EtherCAT master board was mounted on a Vadatech AMC105 PMC/XMC carrier and installed in each MTCA.4 chassis.

Additionally, EtherCAT was also used for magnet power supply control. To reduce the construction cost, a few magnet power supplies were reused that were no longer required at other accelerator in our site. These power supplies had optical remote I/O to connect to an OPT-CC VME board developed at SPring-8 as a field bus in the past. A gateway was prepared for EtherCAT and OPT-CC to accept

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV014

them. All the magnet power supplies were controlled using a PC server with an Advanet AdEXP5712 PCI Express EtherCAT master board.

Eight beam profile monitors were present for adjusting the beam position and shape by watching the screens. Stepper motors were driven to move the screens and adjust their focus. Moreover, the stepper motors were used to move the collimator to adjust the electron beam charge. Their positions were read by resolver counters. A second PC server with an EtherCAT master board was prepared to control the motor controllers and the resolvers.

The number of slaves for six EtherCAT masters is listed in Table 2.

Table 2: The Number of Slaves for Six EtherCAT Masters

Host	Target equipment	Number of slaves
1	linac gun PLC, 238 PLC, 476 PLC, vacuum PLC	4
2	linac S-band PLC, vacuum PLC × 2	3
3	linac C-band1 PLC, C-band2 PLC	2
4	linac C-band3 PLC, C-band4 PLC, vacuum PLC	3
5	linac magnet power supplies	65
6	linac beam profile monitor × 8, collimator, resolver counter	13

GigE Vision Camera

To watch the beam image on the screen of a beam profile monitor, a GigE vision-enabled JAI GO-2400M camera was selected [10]. A PC server was set up and two Neousys Tech PCIe-PoE354at PoE-type network interface boards and a K. K. Rocky RCB-LVDS-TRIG8 trigger input counter board were installed. Eight cameras were directly connected to each port of the PCIe-PoE354at with Ethernet cables. Power supply control via Power-over-Ethernet was also performed. The RCB-LVDS-TRIG8 received the trigger signal and distributed the LVDS standard signal to the eight cameras using an Ethernet cable. Next, the TTL standard signal was input to the camera through the LVDS-TTL translator (RCB-LVDS2TTL).

The Aravis open source library was used for camera control and to read out the image data. The collected image data were stored on an NFS server in the HDF5 format with the associated meta-information such as camera gain and shutter speed.

EQUIPMENT CONTROL OF THE EXISTING STORAGE RING

VME

The storage ring control system consisted of subsystems of RF, magnet, vacuum, monitor, utility, and safety. Each subsystem was controlled using a VME equipped with Solaris 10 OS. As the control equipment for the storage ring was practically unchanged, the same CPU board, I/O board, and chassis were utilized.

A test bench for the RF control was constructed after the shutdown of the NewSUBARU in July 2020. The test bench was prepared to identify the libraries required to transition to a new control system, such as, the gcc4. It was confirmed that the updated equipment control software would work without any challenges. The RF control was chosen for the test because it had the largest number of I/O boards to handle, and only some of them were used in the NewSUBARU.

In the MADOCA framework, the executable files, and the configuration files for the equipment control software running on the VME CPU were placed on a file server. In December 2020, a dedicated file server for the NewSUBARU was installed, and the old mount point was changed to a new one. This allowed easy transfer to the new control system.

For the electron charge measurement system, an interrupt register board was added to collect the data synchronized with the injection trigger signal. The synchronized data acquisition software was prepared using the basic functions of the new control system.

CONTROL FRAMEWORK

The new control framework of the NewSUBARU is the same as the one already in use at the SACLA/SPring-8.

In the framework, a system called the MDAQ is provided as the data acquisition software. MDAQ_POL collects data at fixed time intervals of 1 to 10 s; MDAQ_SYNC collects data synchronized with external triggers of 1 to 20 Hz; MDAQ_IMG collects image data when requested with a maximum repetition rate of 1 Hz; and MDAQ_WFM collects waveform data when requested with a maximum repetition rate of 20 Hz. The data collected by MDAQ_POL and MDAQ_SYNC were saved to an online database using the Cassandra database management system for storage. The image data collected by MDAQ_IMG and the waveform data collected by MDAQ_WFM were saved in a file server with an NFS mount, and their meta-information was saved in the online database. The number of registered signals collected by MDAQ_POL, MDAQ_SYNC, MDAQ_IMG, and MDAQ_WFM were 3,616, 729, 8, and 352, respectively.

To ensure that the NewSUBARU control system was completely independent of the SACLA/SPring-8 control system, a new file server, database servers, and a message server (MQTT broker) were installed in designated areas of the NewSUBARU building.

Moreover, the computer networks that connect the NewSUBARU building and the annex building of the linac were newly prepared.

OPERATION GUI

As the new linac was inaugurated, the NewSUBARU control room was fully operational when compared to the previous setup where the SPring-8 linac operation was separate in the central SPring-8 control room. In the new layout of the control room, the number of operator consoles

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

was increased from two to six. The OS of the consoles was the SUSE Linux enterprise server 15.

Operational applications such as, accessing the log database and handling control messages are essential for maintaining the operational logic and/or sequence. They are usually in the GUI form. The open source version of the Qt5 was selected to create GUIs. The Qt plug-ins and function libraries were prepared in advance, for instance, message exchanges with child processes, forms, graph platforms, basic figure placement plug-ins, and thread-safe programming samples. For the linac, all the GUIs were newly built. For the storage ring, the old GUIs were all transferred to the Qt-based GUIs. The RF control GUI for the C-band is shown in Figure 3.

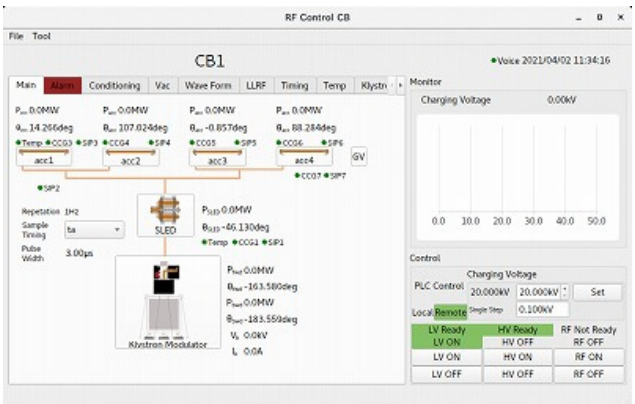


Figure 3: RF control GUI for the C-band.

CONCLUSION

The high-performance dedicated new injector linac has started its operation at the NewSUBARU. This has facilitated the continuation of user experiments after the shutdown of the SPring-8 linac. In addition, the operational flexibility of the NewSUBARU was improved.

To construct the linac, five MTCA.4s were prepared for the RF and monitor, and three PC servers for the magnet power supply and beam profile monitor. Six VMEs of the storage ring were retained with virtually no changes. The maintenance of the VMEs and its replacement may pose a challenge in the future. All the operational GUIs were created using the Qt5. Updating the control system has significantly improved the operation of the NewSUBARU.

Further, the design of the new linac is planned to be used in the 3 GeV synchrotron radiation facility project currently under construction at the Tohoku University campus in eastern Japan [11]. The project plan includes utilizing the same equipment control hardware comprising the MTCA.4, EtherCAT, and GigE camera as well as the same control framework. The project is scheduled to observe its first light in the first half of 2023.

REFERENCES

[1] A. Ando *et al.*, "Isochronous storage ring of the New-SUBARU project", *J. Synchrotron Rad.*, vol. 5, pp. 342–344, May 1998. doi:10.1107/S0909049597013150

[2] T. Inagaki *et al.*, "Construction of a compact electron injector using a gridded RF thermionic gun and C-band accelerator", in *Proc. 12th Int. Particle Accelerator Conf. (IPAC2021)*, Campinas, SP, Brazil, 2021, pp. 2687–2689. doi:10.18429/JACoW-IPAC2021-WEPAB039

[3] R. Tanaka *et al.*, "Control system of the SPring-8 storage ring", in *Proc. 5th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'95)*, Chicago, USA, October 1995, pp. 201

[4] N. Hosoda *et al.*, "Pilot application of new control system at SPring-8 RF test stand", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2017)*, Barcelona, Spain, Oct. 2017, pp. 597–600. doi:10.18429/JACoW-ICALEPCS2017-TUPHA081

[5] K. Okada *et al.*, "Database scheme for unified operation of SACLA/SPring-8", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2017)*, Barcelona, Spain, Oct. 2017, pp. 201–205. doi:10.18429/JACoW-ICALEPCS2017-TUBPA03

[6] T. Sugimoto *et al.*, "Status of the control system for fully integrated SACLA/SPring-8 accelerator complex and new 3 GeV light source being constructed at Tohoku, Japan", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, New York, NY, USA, Oct. 2019, pp. 904908. doi:10.18429/JACoW-ICALEPCS2019-WECPL01

[7] T. Ohshima *et al.*, "Development of a new LLRF system based on microTCA.4 for the SPring-8 storage ring", in *Proc. 8th Int. Particle Accelerator Conf. (IPAC2017)*, Copenhagen, Denmark, May 2017, pp. 3996–3999. doi:10.18429/JACoW-IPAC2017-THPAB117

[8] T. Fukui *et al.*, "Development of the MTCA.4 I/O cards for SPring-8 upgrade and new 3 GeV light source", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, New York, NY, USA, Oct. 2019, pp. 665–670. doi:10.18429/JACoW-IPAC2019-WEPGW029

[9] M. Ishii, T. Takeuchi, C. Kondo, and T. Fukui, "A control system using EtherCAT technology for the next-generation accelerator", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, New York, NY, USA, Oct. 2019, pp. 1258–1261. doi:10.18429/JACoW-ICALEPCS2019-WEPHA068

[10] A. Kiyomichi *et al.*, "Construction of beam monitor control system for beam transport from SACLA to SPring-8", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, New York, NY, USA, Oct. 2019, pp. 1544–1547. doi:10.18429/JACoW-ICALEPCS2019-THAPP03

[11] N. Nishimori, H. Tanaka, and T. Watanabe, "A highly brilliant compact 3 GeV light source project in Japan", in *Proc. 10th Int. Particle Accelerator Conf. (IPAC2019)*, Melbourne, Australia, May 2019, pp. 1478–1481. doi:10.18429/JACoW-IPAC2019-TUPGW035

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

CONTROL SYSTEM OF THE SRILAC PROJECT AT RIBF

A. Uchiyama†, M. Fujimaki, N. Fukunishi, Y. Higurashi, E. Ikezawa, H. Imao, O. Kamigaito, M. Kidera, M. Komiyama, K. Kumagai, T. Nagatomo, T. Nakagawa, T. Nishi, J. Ohnishi, K. Ozeki, N. Sakamoto, K. Suda, T. Watanabe, Y. Watanabe, K. Yamada,
RIKEN Nishina Center, Wako, Japan
A. Kamoshida, National Instruments Japan Corporation, Tokyo, Japan
K. Kaneko, R. Koyama, T. Ohki, K. Oyamada, M. Tamura, H. Yamauchi, A. Yusa,
SHI Accelerator Service Ltd., Tokyo, Japan

Abstract

The RIKEN linear accelerator (RILAC), an injector of the Radioactive Isotope Beam Factory (RIBF) project, was upgraded by installing a superconducting RIKEN linear accelerator (SRILAC) and a 28-GHz ECR ion source (SRILAC project). In addition to controlling these two new apparatuses, the control system of the updated RILAC requires various improvements to the shortcomings of the previous RILAC control system, for example, control methods for electromagnet power supplies using GPIB, a low-performance machine protection system. Moreover, there were issues regarding the integration of small LabVIEW-based systems into the main part of the control system. For efficient operation in the SRILAC project, a distributed control system utilizing the Experimental Physics and Industrial Control System (EPICS) should be adopted in the other parts of RIBF. A higher-level application protocol needs to be integrated into the EPICS channel access protocol. We developed new systems to solve the issues mentioned above and introduced systems that have been proven in other facilities, such as Archiver Appliance as a data archive system. The control system was successfully upgraded and used in the SRILAC beam commissioning completed in 2020. The new control system is currently in its operational phase.

INTRODUCTION

The RIKEN Radioactive Isotope Beam Factory (RIBF) accelerator facility comprises five cyclotrons and two linear accelerators. One of the linear accelerators, the RIKEN linear accelerator (RILAC) [1], has been utilized not only as an injector for cyclotrons but also for stand-alone experiments such as the synthesis of superheavy elements. As an upgrade project, to perform search experiments for superheavy elements with atomic numbers of 119 and higher [2], the superconducting RIKEN linear accelerator (SRILAC) has been introduced downstream of RILAC to enhance beam energy, and a 28-GHz superconducting electron cyclotron resonance ion source (28-GHz SC-ECRIS), similar to the existing RIKEN 28-GHz ECRIS[3], has been introduced at the frontend of RILAC to increase beam intensity [3].

The RIBF control system was constructed using a distributed control system based on the Experimental Physics and Industrial Control System (EPICS) [4]. The new control system for the RILAC upgrade project should be constructed using the EPICS-based control system, and it should be integrated into the existing RIBF control system. The RILAC upgrade project includes various types of hardware such as a 28-GHz SC-ECRIS, beam energy position monitors (BEPM) [5], N2 gas-jet curtain systems [6], and superconducting cavities [7]. Furthermore, the new control system requires improvements to the existing RILAC control system, such as a control method for electromagnet power supplies, a machine protection system, and a data archive system. The protocol for a higher-level application environment should be unified with the EPICS channel access (CA) protocol to enhance operational efficiency. We decided to mainly adopt programmable logic controllers (PLCs) to construct the system and utilize other methods for areas where application of PLCs is difficult considering our limited manpower and resources.

RIKEN 28-GHZ SC-ECRIS CONTROL

The 28-GHz SC-ECRIS was commissioned in advance of the commissioning of other apparatuses. The control system comprises the following: control of ion-source-specific devices (e.g., gas flow controllers, an insertion device of material rods, a high-voltage power supply, and a gyrotron), control of superconducting electromagnet power supplies, and vacuum control [8]. The superconducting electromagnet power supplies embedded MELSEC-Q series as a PLC are connected via NetDev[9], which is an EPICS device support, and controlled by a PC-based EPICS input/output controller (IOC). On the contrary, the control of the ion-source-specific devices and the vacuum control mainly comprised FA-M3 PLCs manufactured by Yokogawa Electric Corporation. These systems have a multi-CPU configuration, a sequence CPU (F3SP71), and a Linux CPU (F3RP61-2L)[10]. Mainly, interlock logic is implemented on the sequence CPU, and EPICS required for higher-level applications is implemented on the Linux CPU.

In contrast to other control systems, some control stations with digital and ana-log modules need to be installed on the high-voltage stage with several tens of kilovolts. Therefore, the connection is insulated between the devices in the high-voltage stage and some devices on the ground level. The control system of the existing 28-GHz SC-ECRIS uses TCP/IP to ex-change interlock signals between

† a-uchi@riken.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the safety system and the control stations in the high-voltage stage. To enhance reliability, the new control system uses two different types of CPUs in the main PLC station, which are connected to five PLC substations with star-topology field bus communication using optical fiber [11]. As a result, the interlock signal also exchanges to the control station on the high-voltage stage through field-bus communication using the optical fiber, and the system reliability was improved compared with the system using the TCP/IP-based interlock signal.

MAGNET POWER SUPPLY CONTROL

The introduction of the 28-GHz ECRIS causes a layout change in the low-energy beam transport (LEBT) system, where new power supplies have been introduced to the electromagnets used. In contrast, several old electromagnet power supplies are used for the electromagnets used in medium-energy beam transport (MEBT) systems. The electromagnet power supplies for MEBT use GPIB as the communication protocol for the control system, and their performance is low compared with recent methods such as IP network-based systems. Therefore, we upgraded the control method of power supplies by replacing the GPIB communication with the VME-based method, which is the standard meth-od of the RIBF control system.

In the VME-based method, electromagnet power supplies are controlled by NIO (NIO) system, as in the case of SPring-8 [12]. The NIO system comprises a master board (NIO-C) and a slave board (NIO-S). Optical fibers connect NIO-Ss to an NIO-C inserted in the VME chassis via a branch board. The EPICS IOC in-stalled in VxWorks controls the object by accessing the shared RAM of the NIO-C, and a maximum of 43 NIO-S can be connected to each NIO-C.

For a high-energy beam transport (HEBT) system, old electromagnet power supplies are also used in the new system. Six of these electromagnet power supplies were remotely controlled by GPIB communication, similar to the MEBT system, and no interlock outputs were implemented for the machine protection system described later. As these power supplies were manufactured more than 20 years ago, converting them to the NIO method, as well as the MEBT electromagnet power supplies, has many limitations and is expensive. Therefore, the GPIB communication boards were removed from the MEBT magnet power supplies and the GPIB communication was modified with an easy-to-handle transistor-transistor-logic (TTL) communication by controlling the IO directly with PLC DI/DO.

The controller utilizes a two-CPU module configuration of F3SP71-4S and F3RP71-2L (see Fig. 1). The sequence CPU implements the core of hardware control, such as BIT output, to an electronic substrate, and a strobe signal timing. The Linux CPU module becomes the EPICS IOC and communicates with the man-machine interface to set the current value and provide the status of the electromagnet power supply. In the beam commissioning, approximately 30 modified electromagnet power supplies were controlled via the EPICS CA protocol without GPIB communication.

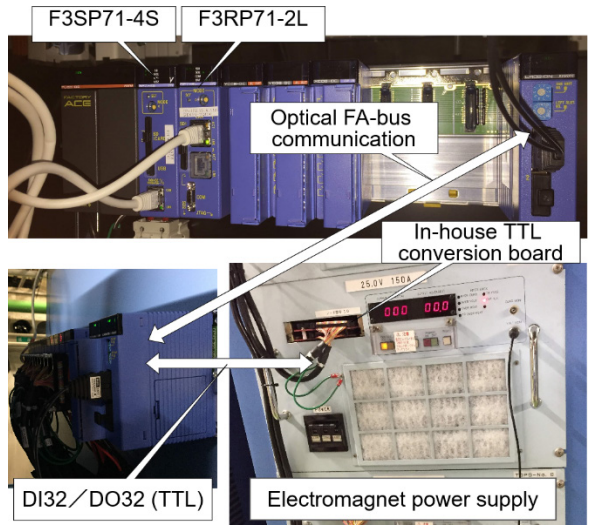


Figure 1: Example of a power supply updated from GPIB to TTL using FA-M3 PLC.

MACHINE PROTECTION SYSTEM

Motivation

In increasing the beam intensity and energy by this upgrade, hardware damage caused by failures of the accelerator components; for example, damage caused by unintended beam orbit changes, will be more serious. As a result, it may cause problems for the beamline components of the HEBT and experimental equipment. In addition, we must prevent problems occurring at the superconducting cavities of SRILAC, such as quenching due to vacuum deterioration caused by beam loss. Therefore, the machine protection system, which is a mechanism to protect the hardware from high-power beams, needs to have a higher performance than before. Before the SRILAC project, the interlock system used in RILAC was realized only via hardware combined with mechanical relays. As a result, the versatility of conventional systems is not high, and hardware circuit modification is required when the input signals are increased, or the logic is changed.

The above-mentioned machine protection system aims to protect hardware from damage caused by beams. In addition, a human protection system is re-quired to secure the radiation safety of individuals involved in accelerator operations and experiments. These two safety systems have different purposes, and systems dedicated to their own purposes should be installed. However, both functions—machine protection and human protection—were implemented in one system, and the wiring and logic were very complicated.

Beam Interlock System

In the RIBF accelerator facility other than RILAC, a beam interlock system (BIS) was introduced in 2006 as a machine protection system. The BIS is a system that stops the beam immediately by exciting a beam chopper that deflects a beam just downstream of the ion sources. The BIS

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

also inserts one of the Faraday cups that is pre-assigned according to the position of the trouble. The BIS uses the MELSEC-Q series as the PLCs [13]. The RIBF BIS comprises five stations. Each station has a CPU module (or modules) and is connected to each other by a MELSECNET/H loop configuration for inter-CPU communication.

System Requirements

In developing a machine protection system for the SRILAC project, the following requirements should be fulfilled [14]:

- Avoid as much complexity as possible in both the hardware and software. Human protection and machine protection should be clearly separated.
- The main part of the system logic should follow the RIBF BIS.
- Select hardware that can be maintained for a long time.
- The entire system development and operation process should be completed within our group to enable system modifications and quick troubleshooting.
- A higher-level system will be developed based on EPICS Channel Access (CA), which is the standard control protocol in RIBF.
- A much faster response is required to protect superconducting cavities than the usual response performance achieved by general PLCs.
- The system is scalable, enabling us to accommodate a sudden increase in interlock signals that are urgently required.

Specification

We decided to use FA-M3 PLCs for the BIS of the upgraded RILAC (hereafter, SRILAC BIS). FA-M3 PLCs have been utilized in many RIBF projects [15] and are widely used in safety systems. Because the number of program steps was observed to be large in the prototype, the F3SP76-7S, the most powerful CPU module at the time of system construction, was adopted as the sequence CPU module. The main sequences of the SRILAC BIS were implemented on the F3SP76-7S, a Linux CPU, F3RP71-2L, was also installed as a means of realizing higher-level systems such as the user interface with EPICS, resulting in a multi-CPU configuration.

Communication

The SRILAC BIS avoids the use of inter-CPU communication, and the main unit with the CPU module installed communicates with seven sub-units via an optical FA bus. In this system, the FA bus network is redundantly configured with an optical fiber loop. Based on the cable length and the number of input points, the I/O refresh time of the FA link was estimated to be approximately 110 μs in this configuration.

System Logic

As shown in Fig. 2, various accelerator components and monitors the input interlock signals to the SRILAC BIS and the SRILAC BIS output signals activating the beam chopper switch. Accelerator operators can set the options of the BIS, such as enable/disable interlock, trigger level of analogue inputs, and holding/unholding alarm status for each input signal according to the operation mode. When an anomaly is detected, the chopper stops the beam and inserts the Faraday cup predetermined according to the position of the anomaly.

Table 1: Number of I/O Signal Points Used in the Main Unit and Seven Sub-units of SRILAC BIS. The numbers without parentheses shows the number of I/Os installed and those in parentheses are the number of I/Os actually used as of October 2021.

	Main	#1	#2	#3	#4	#5	#6	#7
DI	56 (36)	56 (41)	32 (29)	32 (32)	32 (28)	32 (17)	32 (10)	0
DO	56 (8)	56 (2)	32 (4)	32 (6)	32 (0)	32 (0)	32 (2)	0
AI	8 (1)	8 (1)	8 (8)	8 (0)	8 (0)	8 (0)	8 (0)	0

I/Os

In general, PLCs can realize a system with a relatively slow response time of a few milliseconds by using standard input modules and transistor output modules (standard I/Os). On the contrary, to realize a system with a response time faster than 1 ms, a field-programmable gate array (FPGA) is selected [14]. In the case of the SRILAC operation, a fast response time is required to stop the beam when triggered by discharges occurring in superconducting cavities. We selected the FPGA-based high-speed IO module (F3DF01: FPIO module) [16] to implement the logic. It has 24 input points and 24 output points as high-speed I/Os with an embedded Xilinx Spartan®-6 LX FPGA. As the logic can be implemented without going through the sequence CPU module, it can be processed in parallel without being affected by the scan time of the program running on the sequence CPU module side. In this system, each input signal of the FPIO module is enabled/disabled via the internal registers of F3SP76-7S. Table 1 lists the number of inputs and outputs used in SRILAC operation.

Performance

The response speed of standard I/O was measured. The conditions were as follows: an interlock signal was input to the F3XD32-3F of subunit #6 at a 100-Hz repetition rate, and the system response time between the input signal and the corresponding output signal to the chopper switch was measured. We confirmed that the system operated at an average of 6 ms. We also measured high-speed I/O. An interlock signal of 100 Hz was input to the FPIO module installed in the main unit, and the time until the output signal to the chopper switch was measured as approximately 78 μs.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

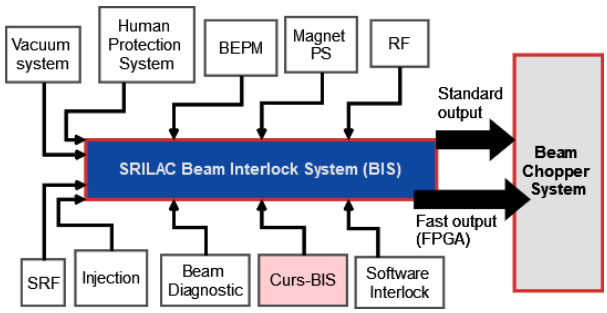


Figure 2: Flowchart of the entire system of machine protection for RILAC operation. The SRILAC BIS is shown as the central blue box. The beam is stopped by the beam chopper by inputting the interlock signals from other systems. There are two outputs—fast and standard.

Curs-BIS

During the SRILAC beam commissioning, a software interlock based on CA was used to detect fluctuations in the excitation currents of the magnet power supplies. Although the software interlock is easy to implement, its reliability and response performance are not high. As the subsystem of RIBF BIS, a beam interlock system driven by a change in current (Curs-BIS) [17] has been developed and used to protect hardware from unintentional changes in the excitation currents of the magnet power supplies. The Curs-BIS has been verified to be effective based on the operation experience of the RIBF BIS up to October 2021, and it has also been introduced to the SRILAC BIS in that all electromagnet power supplies upstream the cryomodules of SRILAC are included. In the near future, Curs-BIS will be installed in all magnet power supplies, including other courses in RILAC.

INTEGRATION WITH SCADA SYSTEM

The MELSEC iQ-R series PLCs are utilized as the controller for monitoring three SRILAC cryomodules, a differential-pumping system, and the readout and settings of the low-level radio frequency (LLRF) system. They work as supervisory control and data acquisition (SCADA) systems. To implement higher-level applications, they should also be integrated into the EPICS.

In the case of EPICS, NetDev as the device support software is adopted to interface the MELSEC iQ-R series with EPICS IOCs, because the MELSEC communication protocol (MC protocol) is a communication protocol with other systems, as in the conventional MELSEC Q series. The IOCs are called virtual-IOCs, which are constructed on a Linux environment with VMware vSphere Standard, and four PLCs for the LLRF system, cryomodules, and the differential-pumping system, are connected to five IOCs, with approximately 5,000 process variables (PVs) in operation.

LABVIEW-BASED SYSTEM

In the control system of the SRILAC project, not only the EPICS IOC running on Linux but also the Windows-based EPICS IOCs are in operation for the BEPM [5] and N_2 gas-jet curtain system [6], and control systems were developed based on LabVIEW. The advantage of the LabVIEW-based system is that the hardware interface is often already provided, and thus development costs can be reduced. On the contrary, LabVIEW-based systems are often operated as two-layer control systems that comprise a PC and devices. We introduced CALab [18] into these LabVIEW-based systems as a SoftIOC, which behaves as a Windows IOC. As a result, these Lab-VIEW-based systems have been successfully integrated into other EPICS-based systems.

HIGHER-LEVEL APPLICATION

Data Archive System

With the number of archived data points expected to increase dramatically owing to the operation of SRILAC, the Archiver Appliance was deployed to improve the data archiving performance [19]. Since 2009, RIBF has been using the RIBF control archive system (RIBFCAS) developed by RIKEN Nishina Center as a data archive system for the data generated by the EPICS IOCs [20]. In contrast, MyDAQ2, developed at SPring-8 as a data store, was also introduced mainly for non-EPICS-based systems [21]. In the RIBF control system, MyDAQ2 is utilized not only as a data store, but also as a method to read data from a non-EPICS-based system through an EPICS CA without EPICS device support software [22]. To realize a user-friendly system for data visualization, the data of RIBFCAS, MyDAQ2, and Archiver Appliance should be visualized on the same system. Therefore, we herein developed a system that implements a Web API to convert RIBFCAS and MyDAQ2 data into the JSON format, thereby making it possible to unify the data format with the Archiver Appliance and display the data with the same viewer software. The maximum data-acquisition cycle was improved to 10 Hz, compared to a cycle of 1 to 20 s in conventional systems.

Operator Interface

To develop the operator interface (OPI) for SRILAC, we use Control System Studio (CSS) as well as MEDM/EDM, which we have been using for a long time. The developed RF control screen for the SRILAC is shown in Fig. 3. In the case of MEDM/EDM, the GUIs are developed on a server installing CentOS7.7, and the developed GUIs are displayed on the screen of each local machine by X11 forwarding. Therefore, the development and execution of the GUIs are performed on the same server. On the contrary, in the case of CSS, running multiple instances on a server is difficult; therefore, CSS is installed separately in each local environment. At this time, we introduced ownCloud [23] as a means of sharing the most recent OPI files with the executors. ownCloud is a system that allows us to introduce online storage such as Google Drive through an intranet and is effective in overcoming the complexity of file sharing using NFS, wherein detailed access control of OPI files to users is difficult.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

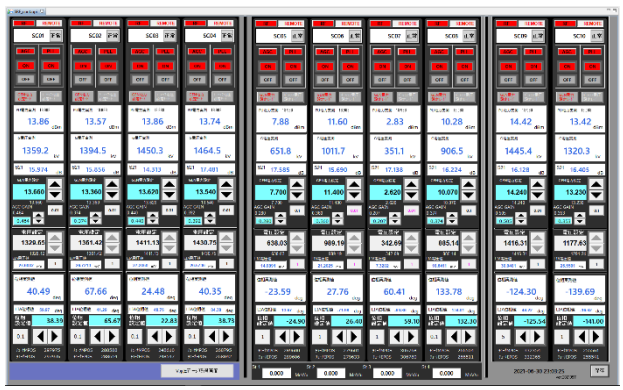


Figure 3: Screenshot of superconducting radio frequency cavities control panel developed by CSS.

CONCLUSION

New control systems have been developed for components newly introduced in the SRILAC project. In addition, several improvements have been made to solve the limitations of controlling the existing components used in RILAC. The development and implementation of the SRILAC BIS were successfully completed. The anomaly signals from all magnet power supplies, RF systems, and vacuum gate valves are input to the SRILAC BIS, and the beam can be immediately stopped to protect the hardware by triggering the beam chopper. The integration of the LabVIEW-based system into EPICS, which had been a serious issue in the past RIBF control system, was also accomplished, and higher-level applications can now be developed in a unified manner using the CA protocol. As a result of these efforts, beam commissioning was successfully completed in 2020.

ACKNOWLEGEMENT

The authors would like to thank Dr. Rok Hrovatin in Cosylab for technical advice with development of the machine protection system.

REFERENCES

[1] M. Odera I, "Variable frequency heavy-ion linac, RILAC: I. Design, construction and operation of its accelerating structure", *Nucl. Instrum. Methods A*, vol. 227, p. 187, 1984. doi:10.1016/0168-9002(84)90121-9

[2] N. Sakamoto *et al.*, "Development of Superconducting Quarter-Wave Resonator and Cryomodule for Low-Beta Ion Accelerators at RIKEN Radioactive Isotope Beam Factory", in *Proc. 19th Int. Conf. RF Superconductivity (SRF'19)*, Dresden, Germany, 2019, pp. 750-757. doi:10.18429/JACoW-SRF2019-WETEB1

[3] T. Nagatomo *et al.*, "High intensity vanadium beam for synthesis of new superheavy elements with well-controlled emittance by using 'slit triplet'", *Rev. Sci. Instrum.*, vol. 91, p. 023318, 2020. doi:10.1063/1.5130431

[4] M. K. Fujimaki, M. Kase, A. Uchiyama, and M. Komiyama, "Status of Control System for RIKEN RI-Beam Factory", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper TPPB11, pp. 187-189.

[5] T. Watanabe *et al.*, "Commissioning of the Beam Energy Position Monitor System for the Superconducting RIKEN Heavy-ion Linac", *in Proc. 9th Int. Beam Instrumentation Conf. (IBIC'20)*, Santos, Brazil, Sep. 2020, pp. 295-302. doi:10.18429/JACoW-IBIC2020-FRAO04

[6] H. Imao *et al.*, "He gas stripper with N2 gas-jet curtain", RIKEN Accel. Prog. Rep. 52, 2019, pp. 14.

[7] K. Yamada *et al.*, "Construction of Superconducting Linac Booster for Heavy-Ion Linac at RIKEN Nishina Center", *in Proc. 19th Int. Conf. RF Superconductivity (SRF'19)*, Dresden, Germany, Jun.-Jul. 2019, pp. 502-507. doi:10.18429/JACoW-SRF2019-TUP037

[8] A. Uchiyama *et al.*, "Control System for the New RIKEN 28-GHz Superconducting Electron Cyclotron Resonance Ion Source for SRILAC", *Rev. Sci. Instrum.*, vol. 91, 2020, p. 025101, 2020. doi:10.1063/1.5129632

[9] J. Odagiri *et al.*, "EPICS Device/Driver Support Modules for Network-based Intelligent Controllers", *in Proc. 9th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'03)*, Gyeongju, Korea, Oct. 2003, paper WP563, pp. 494-496.

[10] EPICS Device and Driver Support for Yokogawa's F3RP71 and F3RP61, https://github.com/EPICS-F3RP61/epics-f3rp61

[11] A. Uchiyama *et al.*, "Design of Reliable Control with Star Topology Fieldbus Communications for an Electron Cyclotron Resonance Ion Source at RIBF", *in Proc. 12th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'18)*, Hsinchu City, Taiwan, Oct. 2018, pp. 105-108. doi:10.18429/JACoW-PCaPAC2018-WEP30

[12] S. Ueda *et al.*, "Upgrade of the control system for steering magnet power supplies at SPring-8 booster synchrotron", in *Proc. the 9th Annual Meeting of Particle Accelerator Society of Japan*, Toyonaka, Japan, Aug. 2012, pp. 694-697.

[13] M. Komiyama, M. Fujimaki, N. Fukunishi, A. Uchiyama, M. Hamanaka, and T. Nakamura, "Recent Update of the RIKEN RI Beam Factory Control System", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 427-430. doi:10.18429/JACoW-ICALEPCS2017-TUPHA028

[14] R. Schmidt, "Machine Protection and Interlock System for Large Research Instruments", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 537-542. doi:10.18429/JACoW-ICALEPCS2015-TUC3I01

[15] M. Komiyama, M. Fujimaki, N. Fukunishi, J.-I. Odagiri, and A. Uchiyama, "Upgrading the Control System of RIKEN RI Beam Factory for New Injector", *in Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper TUP084, pp. 275-277.

[16] Yokogawa Electric Corporation, https://partner.yokogawa.com/global/itc/index.htm

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[17] K. Kumagai *et al.*, "Development of beam interlock system driven by change in current of the magnet", RIKEN Accel. Prog. Rep. 52, 2019, pp. 115.

[18] CA Lab (LabVIEW + EPICS) - Helmholtz-Zentrum Berlin (HZB), https://www.helmholtz-berlin.de/zentrum/locations/it/software/exsteuer/calab/index_en.html

[19] A. Uchiyama, N. Fukunishi, M. Kidera, and M. Komiyama, "Data Archive System for Superconducting RIKEN Linear Accelerator at RIBF", in *Proc. 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, Brazil, May 2021, pp. 2178-2181. doi:10.18429/JACoW-IPAC2021-TUPAB297

[20] M. Komiyama, N. Fukunishi, and A. Uchiyama, "Construction of New Data Archive System in RIKEN RI Beam Factory", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper MOPKN005, pp. 90-93.

[21] T. Hirono, T. Matsushita, T. Ohata, and A. Yamashita, "Development of Data Logging and Display System, MyDAQ2", in *Proc. 7th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'08)*, Ljbuljana, Slovenia, Oct. 2008, paper TUY01, pp. 55-57.

[22] A. Uchiyama, M. Fujimaki, N. Fukunishi, and M. Komiyama, "Integration of Standalone Control Systems into EPICS-Based System at RIKEN RIBF", in *Proc. 11th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'16)*, Campinas, Brazil, Oct. 2016, pp. 35-37. doi:10.18429/JACoW-PCAPAC2016-WEPOPRPO12

[23] Daniel van der Ster and Arne Wiebalck, "Building an or ganic block storage service at CERN with Ceph", *J. of Physics: Conference Series*, vol. 513, 2014, p. 042047. doi:10.1088/1742-6596/513/4/042047

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV016

DESIGN AND IMPLEMENT OF WEB BASED SCADA SYSTEM FOR HUST FIELD-REVERSED CONFIGURATION DEVICE

F. Wu†, Y. Jiang, S. Li, B. Rao, W. Wang, X. Xie, Y. Yang, M. Zhang, P. Zhang, W. Zheng
International Joint Research Laboratory of Magnetic Confinement Fusion and Plasma Physics,
State Key Laboratory of Advanced Electromagnetic Engineering and Technology,
School of Electrical and Electronic Engineering,
Huazhong University of Science and Technology, Wuhan, 430074, China

Abstract

As a large complex fusion research device for studying field reversed configuration (FRC) plasma, HUST FRC (HFRC) is composed of many subsystems. In order to coordinate all systems and ensure the correct, orderly and stable operation of the whole experimental device, it is very important to have a unified and powerful control system.

HFRC SCADA (Supervisory Control And Data Acquisition) system has selected the in-house developed CFET (Control system Framework for Experimental Devices Toolkit)as the control framework, with advantages of strong abstraction, simplified framework, transparent protocol and flexible extension due to Web technology.

Introduction

Growing worldwide demand for energy, and problems of scarcity and environmental impact associated with conventional sources are the challenges facing humanity [1]. The energy industry is facing decades of transformation. On a longer scale, nuclear fusion will be part of a catalog of more sustainable energy sources [2].

Field-reversed Configuration (FRC) has the advantages of complete axis symmetry, relatively simple structure, β~1 and so on [3]. Therefore, it is of great significance for the research of new fusion configuration, and the future exploration of miniaturization and economization of the fusion reactors [4, 5]. HUST Field-Reversed Configuration (HFRC) is a pre-research platform for field-reversed configuration research device based on plasma's colliding and merging, which is under development.

HFRC is composed of many subsystems such as magnets, vacuum, power, and diagnostics. It's important to have a unified control system considering the diversity and complexity of the subsystems. This paper describes the design and implementation of control system specially designed for HFRC.

HFRC SCADA (Supervisory Control And Data Acquisition) system includes the global control system, which coordinate all subsystems, and control systems in various subsystems such as the pulse power supply control system. It also offers a customizable and pluggable web based HMI for better operation. Users can design the HMI by simply dragging and dropping widgets in the browser.

At present, HFRC SCADA has been initially deployed in daily experiments of HFRC, which will provide valuable experiences for future control system design of large experimental device.

Design of HFRC SCADA

Micro Service HFRC SCADA adopted the decentralized model on the whole, and it can mainly be divided into two parts: central control system and subsystems. The main function of central control system is to coordinate the run of whole experimental device, and all subsystems need access plant operate network and accept the monitoring and the dispatch from central control system. The main subsystems are charger control system, pulse power control system, central timing control system, distributed timing control system and data acquisition system. Each subsystem offers different service to others. Fig.1 shows the overall structure of the whole system.

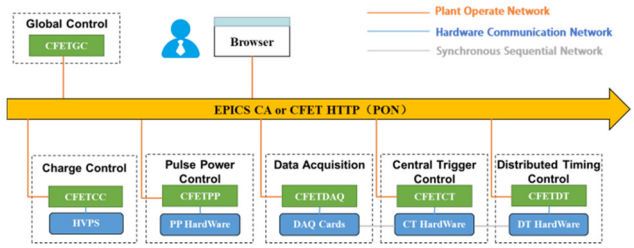


Figure 1: Overall structure of the whole system.

HFRC SCADA builds the whole system as a suite of small services, each running in its own process and are independently deployable. Adopting micro services makes HFRC SCADA simpler to deploy and understand, and minimizing the risk of change.

FSM (Finite State Machine) Pattern The discharge process of HFRC can be seen as a sequential transitions of states. HFRC SCADA uses FSM mode to control the flow of discharging and handle exceptions.

FSM is reliable, easy to understand, predictable and safe. There is only one state active at any one time, which drastically reduces the chance of unforeseen errors or unexpected behavior in the system.

Observer Pattern In observer pattern, any other object can be registered on an observed object, which will function as an observer. The first object, also called the subject, informs registered observers each time it is modified [6].

As HFRC SCADA is a distributed event handling system, observer pattern works well in implementing coordination of subsystems. Each subsystem can act as both an observer and a subject.

* Work supported by No. 2017YFE0301803 and No.51821005
† email address: wfy@hust.edu.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV016

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CFET HFRC SCADA selects the in-house developed CFET (Control system Framework for Experimental Devices Toolkit) as the control framework. CFET is a general software framework for fusion device integrated control system development based on Web technology, and can be used as supervisory control and data acquisition (SCADA) software.

Based on the technology of Web, CFET adopts HTTP protocol and RESTful design principle as the general communication protocol of the system. Under the CFET framework, the object realizing the same business function logic is called as a Thing. As is shown in Fig. 2, all resources in CFET system can be accessed by an URL through the internet. It is just the same as the URL we meet in the browser and someplace else. The head of the under UDA is the address of the CFET Host and in the end is the query string. One URL can locate anresource of a CFET Thing in a CFET Host of a CFET CODAC system [7].

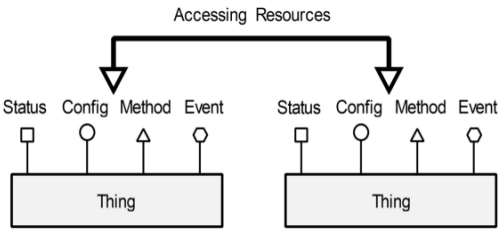


Figure 2: CFET abstract model.

Under the CFET framework, each subsystem can access plant operate network and realize the function of coordination and monitoring.

Hardware Module

The most commonly used field-reversed plasma formation method is based on the Field-Reversed Theta-Pinch (FRTP) formation [8]. HFRC device is mainly divided into plasma formation area, collision fusion and compression area. The FRTP is located in the formation area and is mainly used for the formation and injection of FRC plasma. The control system controls the discharge process mainly by orchestrating the power supply in the formation area.

In order to obtain the desired results, HFRC has strict requirements on the discharge timing. If the trigger timing deviates more than expected, the obtained current waveform will not match the expected one, and the result will be unsatisfactory. Considering the adaptability to the CFET framework and the problem of generating the trigger timing signal with high accuracy, the formation area power controller is developed independently.

As is shown in Fig. 3, formation area power controller hardware mainly includes MCU real-time slow control board, FPGA trigger board, AC-220V to DC-12V power supply, three-terminal interface and 1U Standard Chassis.

MCU real-time slow control board is responsible for the generation of slow control signals, such as heating signals, temperature signals. FPGA trigger board generates high-precision trigger signals.

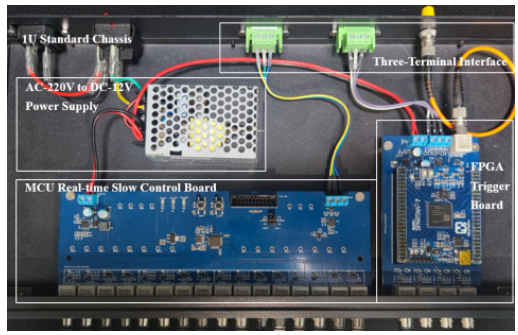


Figure 3: The formation area power controller.

Data Acquisition, Archive and Access

Data acquisition subsystem adopts CFET as framework. CFET's high robustness and low coupling performance allow the subsystem to be deployed in HFRC and other systems with data acquisition requirements.

Data acquisition subsystem is compatible with common formats of data storage and provides full support for files in formats such as HDF5. Both data file and MDSplus are supported to upload experimental data. The MongoDB database is used as the underlying software for the experimental file data access.

HFRC data acquisition subsystem offers different ways of accessing data with full functionality, simplicity and high performance.

HMI

HFRC SCADA implements HMI based on Web technology. Users only need a browser to supervise control and acquire experimental data. HMI also supports UI customization, which means users are able to design their own interfaces. Figure 4 shows the current global control interface.

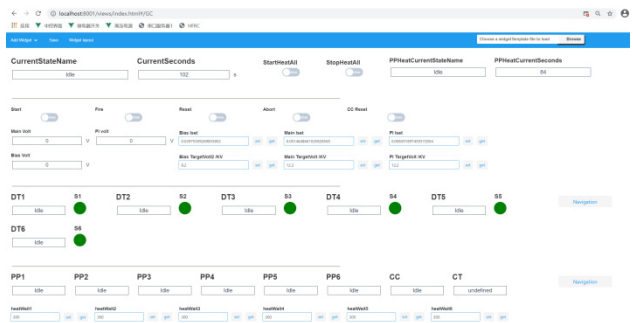


Figure 4: The global control interface.

CONCLUSION

This paper introduces the design and implement of Web Based SCADA System for HFRC in terms of the overall design of the system, hardware module, data acquisition and HMI. HFRC SCADA is able to orchestrate all the instruments and devices to conduct a successful experiment, as well as collect and archive the scientific data generated during the experiments. It also offers a customizable and pluggable web based HMI for better operation.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV016

ACKNOWLEDGEMENTS

The authors are very grateful for the help of J-TEXT team. This work is supported by the National Magnetic Confinement Fusion Science Program (No. 2017YFE0301803) and by the National Natural Science Foundation of China (No. 51821005).

REFERENCES

[1] C. López, "Current Challenges in Energy" in Frontiers of Knowledge, Madrid: BBVA. 2008. https://www.bbvaopenmind.com/en/articles/current-challenges-in-energy/

[2] R. L. Spencer and D. W. Hewett, "Nuclear fusion: Energy for centuries to come…", *Fuel and Energy Abstracts*, vol. 36, pp. 194-194, 1995. doi:10.1016/0140-6701(95)80433-1

[3] R. L. Spencer and D. W. Hewett, "Free boundary field - reversed configuration (FRC) equilibria in a conducting cylinder", *The Physics of Fluids,* vol. 25, pp. 1365-1369, 1982. doi:10.1063/1.863901

[4] S. Zhanga, T. P. Intrator, G. A. Wurden, *et al.*, "Confinement analyses of the high-density field-reversed configuration plasma in the field-reversed configuration experiment with a liner", *Physics of Plasmas*, vol. 12, p. 052513, 2005. doi:10.1063/1.1899648

[5] F. F., Chen, "*An Indispensable Truth: How Fusion Power Can Save the Planet*", 2011, Springer Verlag, doi:10.1007/978-3-319-14409-2

[6] G. Zhai, H. Zheng, and B. Zhang, "Observer-based control for the platform of a tethered space robot", *Chinese Journal of Aeronautics*, vol. 31, pp. 1786-1796, 2018. doi:10.1016/j.cja.2018.01.012

[7] W. Zheng, Y. Wang, M. Zhang, Z. Yang, and Y. Pan, "Designing CODAC system for tokamaks using web technology", *Fusion Eng. Des.*, vol. 146, pp. 2379–2383, 2019. doi:10.1016/j.fusengdes.2019.03.195

[8] J .M. Taccetti *et al.*, "FRX-L: A field-reversed configuration plasma injector for magnetized target fusion", *Rev. Sci. Instrum.*, vol. 74, pp. 4314-4323, 2003. doi:10.1063/1.1606534

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

CERN SCADA SYSTEMS 2020 LARGE UPGRADE CAMPAIGN RETROSPECTIVE

Lukasz Goralczyk†, Alexandros Foivos Kostopoulos, Brad Schofield, Jean-Charles Tournier
CERN, Geneva, Switzerland

Abstract

In this paper we report the experience from a large-scale upgrade campaign of SCADA control systems performed during the second LHC Long Shutdown at CERN. Such periodical upgrades are dictated by the ever evolving SCADA WinCC OA system and the CERN frameworks evolution used in those control systems. These upgrades concern: accelerator control systems, e.g. quench protection system, powering interlocks, magnet alignment; control systems devoted to accelerator facilities such as cryogenics, vacuum, gas and other global technical infrastructure systems as well as the CERN electrical distribution system.

Since there are more than 200 SCADA projects covering the CERN accelerator complex and technical infrastructure, any disruption requires careful coordination, planning and execution with process owners. Having gained experience from previous campaigns and reaching a new level of automation we were able to make visible improvements by shortening the required time and reducing the personnel required. Activities, lessons learned and further improvements are presented as well as a comprehensive statistical insight of the whole campaign.

INTRODUCTION

Software upgrades are an inherent process in modern computing which also applies for present-day SCADA (Supervisory Control And Data Acquisition) applications and the infrastructure in which they run. Upgrades are a necessity to keep systems secure, provide new features and avoid software obsolescence.

At CERN there are numerous industrial control systems in different domains such as machine protection, cryogenics, cooling and ventilation and the electrical distribution system. Together they account for a total of around 200 SCADA applications. These applications are often critical for the operation of CERN's accelerator complex and experiments. The large number and broad scope of these applications poses several challenges with respect to upgrades, a key challenge being the creation of an upgrade plan which does not interfere with the operation schedule.

At the end of 2018 CERN entered its second Long Shutdown (LS2) period that normally lasts around two years (in 2020 due to COVID-19 pandemic, it was extended). This was a perfect opportunity to perform a large-scale campaign to upgrade SCADA applications to the latest software packages, as well as to perform some housekeeping activities (e.g. server maintenance). In most cases, operational constraints are not as strict as is normally the case outside of long shutdown periods.

In this paper we give an insight into the upgrade campaign that took place in 2020. We start by giving details about the motivation behind the upgrades, planning goals and the scope. Differences with the previous campaign, carried out in 2017, will be also discussed. Improvements implemented from that campaign, as well as new enhancements are a topic of another chapter. We follow by presenting details about our experiences – the good and the bad. We conclude with proposals for further improvements.

For details about the SCADA service at CERN as well as software and infrastructure details, the reader is referred to [1].

MOTIVATION, SCOPE AND PLANNING

For the 2020 campaign the motivation was:

1. Siemens/ETM has decided to stop providing support for the version 3.15 of WinCC OA [2], the software used for most of the SCADA applications at CERN. It was, therefore, necessary to upgrade all the CERN WinCC OA-based SCADA systems to the latest version of WinCC OA.
2. New features that came with the updated JCOP [3] and UNICOS frameworks [4] depended on newer releases of WinCC OA.
3. Relaxed intervention restrictions given by the LS2 period.

Planning activities started several months before LS2 began. Similarly to the 2017 upgrade campaign all control domain representatives were interviewed to gain information about the planning restrictions, acceptable downtime and any additional requests. It was also an opportunity to present how the upgrades would be executed.

With this data gathered, an initial schedule was proposed and later agreed upon with the representatives. The upgrade calendar had to take into account some restrictions as: criticality, overlap with other upgrades, operation schedule and availability of experts.

Despite the long duration of LS2, the initial assumption was to finish all upgrades in less than 6 months, with the majority of applications upgraded in the first quarter of 2020. The COVID-19 pandemic forced us to put all pending upgrades on hold, abandon the initial schedule and, later, adjust it again conditioned by the COVID-19 protective measures (e.g. teleworking). This resulted in the upgrades being spread throughout the year.

† lukasz.goralczyk@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

The initial scope of the upgrade was around 250 applications, but with some major improvements and the consequent simplification in the number of applications in one of the largest control project at CERN, the Quench Protection System, the scope was reduced to around 200.

COMPARISON WITH 2017 CAMPAIGN

There are several differences when compared with the previous SCADA applications upgrade campaign; these are summarized in Table 1.

Due to the relatively minor number of changes in the new WinCC OA version, the JCOP and UNICOS frameworks did not require as many adaptations as in the 2017 campaign. This allowed for a shorter testing period. One major change was the switch from the ISO8859-1 character encoding to UTF-8. Care had to be taken to ensure that the upgraded applications text was displayed correctly, as well as not to cause any internal problems.

Table 1: Upgrade Campaign Comparison

Campaign	2017	2020
WinCC OA	3.11 → 3.15 major change	3.15 → 3.16 minor change
Character encoding	ISO8859-1	UTF-8
Operating system	SLC6[1] → CC7	CC7 minor version
Timing and planning	EYETS[2] strict	LS2 relaxed
Resources	7 persons	4 persons
Tooling	Graphical user interface (GUI)	Command line interface (CLI)

In the previous campaign, a major change in the operating system (OS) was a hurdle as it was a considerable change, significantly extending upgrade time, requiring extra staff and a notably larger validation period. The campaign in 2020 had only a minor OS version change and its validation with a newer WinCC OA software was shorter and smoother.

The LS2 period was longer than EYETS (End of the Year Extended Technical Stop) allowing for a more relaxed planning, although the initial assumption was to finish the upgrades in the first half of 2020, with the majority of them in the first quarter. It's worth noting that despite the fact that a number of industrial installations were not operational, many of their control systems were still in production and required utmost care in planning and execution.

It's important to mention that other aspects of the process have not changed; these include:

1. Upgrade schedule to strictly follow directives from control domain representatives.
2. Individual upgrades execution composed of many individual steps: around 70.
3. Close cooperation with controls representatives to ensure proper operation after the upgrade.

[1]Scientific Linux CERN

[2]Extended Year-End Technical Stop

TOOLING IMPROVEMENTS

New tooling around the upgrade process was built on the previous collection of Python and bash scripts and libraries that interacted with WinCC OA.

Two new main tools were developed: *wccadm,* a Python 3 library to interact with WinCC OA and *wccauto,* a simple Python 3 framework to execute predefined steps. *wccadm* substituted a previous, similar, library which was based on unsupported Python 2. It aims to be a more reliable and complete interaction environment for WinCC OA. A major effort was made to make it easy to develop with as well as to have a richer logging support. Well formed and informational (debug) messages turned out to be crucial in the validation and, later, in the execution phase of the upgrade.

wccauto is an environment to develop specialised modules and execute them in a predefined order. Each module implements one particular function, which can be something as simple as displaying a message or, a more complex functionality like saving the SCADA application health status. *wccauto* is also an execution environment, where a predefined set of steps (modules), called plans, are run in an orderly manner (examples of steps could be 'display a message', 'copy a file', 'start the application'). It provides a command line interface and collects detailed logs from the upgrade. A typical production plan consisted of around 70 steps. Depending on the needs, a plan could be adjusted for non-standard applications.

The new tools exclusively use a command line interface (CLI); this differs from the previous approach which provided a graphical user interface (GUI). The switch simplified the execution as only a secure shell (ssh) connection was needed. It also allowed automatic offline tests performed on backups of production SCADA applications; a special non-interactive plan was used for that purpose.

Some of the upgrade steps that previously had to be executed manually were automated, leaving only a handful of operations requiring attention. The majority of the steps would execute automatically one-by-one. The tools would always provide feedback messages of what is being currently done. In case of a problem, a clear error message would be displayed with, if possible, a proposed solution. A number of pre-flight checks were done before the upgrade to detect problems early and save precious time later.

In addition to *wccadm* and *wccauto*, some supplementary programs were created including one to take and compare screenshots of synoptic views before and after the upgrade. This tool was used to detect differences in panel rendering between the old and the new version of WinCC OA, including problems related to the character encoding switch from ISO8859-1 to UTF-8. At least one bug in a panel widget, used by many applications was detected and fixed, as well as multiple

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

panels in the PSEN application, used to supervise CERN's electrical network.

Each upgrade would produce a detailed log that would be recorded in the ticketing system. It also provided a source for post-upgrade analysis of the process. Bottlenecks could be identified and further improvements could be proposed.

CHALLENGES

Such a large scale upgrade, spanning across many control domains, poses several challenges, both at the preparation phase as well as during the execution period.

The creation of the schedule started by meeting with all control domain representatives. The purpose was to present early enough the upgrade, explain the impact on the supervision layer and the availability of the control system to the operation team and learn if there were any operational restrictions and when the upgrade should not be done.

After collecting this initial data a draft of the schedule was created. It grouped SCADA applications by their domains and planned them such that all of them would be done before mid of 2020. Any restrictions and particular wishes were taken into account. Special gaps were introduced to allow rest time for the team and as a safety net in case of unforeseen problems. Naturally the schedule would evolve with time as teams would make adjustments.

One serious, unforeseen external event that interfered with the schedule was the start of the COVID-19 pandemic. At the very beginning, as events quickly developed, we decided to put planned upgrades on hold (with some exceptions). Later we cancelled the current schedule and changed the modus operandi to a more ad-hoc planning. In the end a new schedule was created and agreed with representatives for the remaining SCADA applications. Due to lockdown and the resulting restrictions of on-site presence at CERN, it became clear that it would be necessary to perform upgrades remotely. Initially we approached this with great caution. With time we gained confidence and could resume the upgrades, doing them almost exclusively remotely. See Fig. 1.

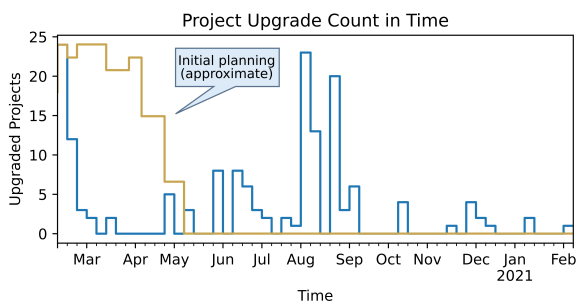


Figure 1: Comparison of upgrade counts in time between initial and adjusted planning.

At any point in time the schedule had to take into account fewer staff availability compared to the previous campaign. It was desired that the load should not reach 100% in order to give time for other parallel activities like support, operations or other developments. Thanks to the improved tooling and documentation the process was not as taxing as with the previous upgrade. For numerous applications the upgrade could run mostly in the background.

WHAT WENT WELL

Despite changes in the schedule and fewer committed resources, we succeeded in upgrading all the SCADA applications.

Developed tools and procedures proved to work well and be very reliable. They provided an intuitive interface with clear instructions on what to do and displaying the currently executed operation. In case of problems, a clear and understandable message would be displayed. Complexity of the process was well hidden from the user. Even after a longer break there was no need for an earlier procedure review as following the tool simple instructions was enough. It is important to mention that the same procedure (a plan, which as mentioned before, is a series of simple steps to be executed) was applied to the majority of the upgrades, thus avoiding possible confusion from constant changes. Parallel upgrades were possible and performed whenever possible thus greatly reducing the impact on supervision unavailability.

The ability to easily change the upgrade plan allowed us to adopt the process to non-standard SCADA applications (see Fig. 2). Additionally any reports of possible improvements would be easily accommodated in the upgrade plan.

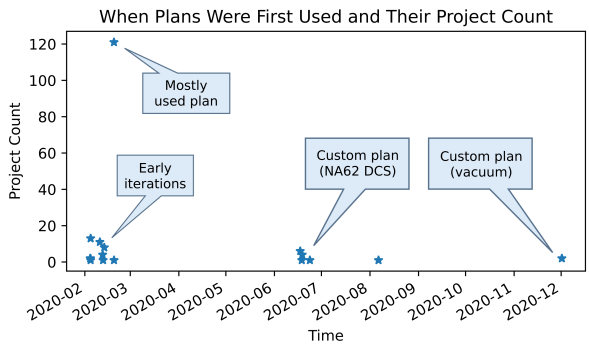


Figure 2: The number of projects using an upgrade plan (vertical axis) vs the time each plan was first used.

A key factor to the success of the campaign was the massive offline testing of the upgrade procedure done before the execution phase. It gave us the confidence especially needed for newly developed tools. A number of problems were detected thanks to detailed logs collected during the test run.

The announced total time needed for a single application upgrade was set to 4 hours initially, based on the experience of the 2017 campaign, but was then reduced to 2 hours, due to improvements made to the tooling. After the expected initial learning phase, the time was further reduced to 1 hour. See also Fig. 3.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

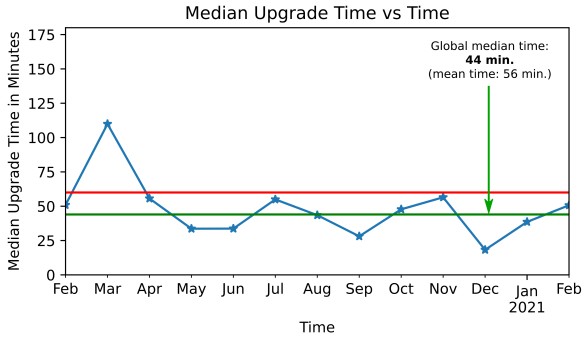


Figure 3: How upgrade time changed throughout the whole campaign.

The LS2 period gave us the comfort of a more relaxed upgrade schedule. Despite ongoing interventions we could still find time to provide support and respond to other client requests.

Finally, the rich logs produced during the upgrades gave us an interesting insight on the whole process, such as where the bottlenecks were, and what the most common errors were in the process. The analysis will prove useful in indicating what to improve in the future.

WHAT DID NOT GO WELL

Not everything went as planned and there were a couple of obstacles we needed to overcome. This section presents the lessons learned from events that didn't go well.

The execution planning was done using an Excel spreadsheet. This was initially a good choice because of its simplicity and ability to view it online using existing infrastructure. Data present in the spreadsheet was later used to create tickets in a tracking system (Atlassian Jira [5]), to enable traceability and integration with a shared calendar (Outlook). This combination was not well suited for constant, large changes. Applying changes to a selected set of tickets was time consuming. A solution to this problem would be to use a dedicated tool that would interact with the ticketing system and a shared calendar. It could also automatically create an initial schedule given some basic rules.

Despite the effort put into the automation of the process, manual steps still existed. List of top 4 slowest steps can be found in Table 2, majority of them is manual. Their automation was either not possible or too costly in terms of time. Automation not only saves time, but also prevents mistakes – we experienced that a couple of times. After the upgrades were completed some of those manual steps were automated. For changes in external systems we contacted other teams to help us with the automation.

In several instances, multiple applications were upgraded in parallel. This was due to various reasons, like a request from the control domain representative or the fact that applications hosted on the same server must be upgraded at the same time. In most cases, this parallelism saved time. It was, however, very onerous for the person performing the upgrade, and could negatively affect the timing, even at the presence of any single problem. The lesson learned was to put a limit on the number of parallel upgrades handled per person; this could be an input for a future automatic scheduling tool.

Table 2: Top 4 Slowest Upgrade Steps (Median)

Upgrade Step	Manual	Time in Minutes
Update of application start-tup scripts	Yes	6
Pre-upgrade scripts	Yes	3.5
Post-upgrade backup	Yes	3.5
JCOP and UNICOS components upgrade	No	2.5

We also encountered a problem with the SCADA software we use – WinCC OA. Despite extensive testing and validation, a few weeks after the first applications were upgraded, a memory leak caused a crash. The problem manifested itself under very specific conditions. Upgrades were temporarily put on hold and a fix was quickly provided and applied to the already upgraded applications. Intentional gaps in time, that were strategically inserted in the planning from the beginning, helped mitigate the impact of various unexpected delays on the overall schedule.

FUTURE IMPROVEMENTS

After the last application was upgraded we already had a good idea about future improvements.

The key to further reduction in the upgrade time is the automation of the remaining manual steps. This will also open the door to an interaction-less upgrade. The revert (start from scratch) operation should also be automated – in specific cases, when an upgrade needed to be repeated, this was done manually.

The most natural evolution of our tools would be to incorporate Ansible [6], an IT automation solution, which perfectly fits our needs. It provides, out of the box, the functionality of *wccauto* and much more. Rich features and plugins can make the process even more resilient to problems and scale better.

These advancements in the tooling should allow us to do offline upgrades, a method in which the process is performed outside of the production system. This method was previously proposed in [1]. The planned use of SCADA applications inside containers could give us similar opportunities.

We should also consider using a dedicated scheduling software, that, based on resource constraints, will automatically create/adjust the planning of the upgrades. This could be either an in-house developed or an existing commercial or open-source software, and it would be a great aid during the initial phase as well as later in the campaign, when changes are unavoidable.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONCLUSION

The CERN SCADA systems upgrade in 2020 followed a steady incremental progress. Flexibility and adaptability in the planning were needed in order to accommodate the operational restrictions posed by the different domains.

The effort placed on automation paid off, by enabling more efficient, agile and error-free upgrades.

The success of the campaign was visible on the overall decreasing intervention times per upgrade and, in particular, on the less human resources needed. The improvements introduced in the tooling were essential to reach this achievement.

The experience gained during the previous campaign in 2017 constituted the base of this accomplishment.

REFERENCES

[1] R. Kulaga et al., "Large-Scale Upgrade Campaigns of SCADA Systems at CERN - Organisation, Tools and Lessons Learned", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 1384-1388. doi:10.18429/JACoW-ICALEPCS2017-THPHA021

[2] Simatic WinCC Open Architecture (previously PVSS) SCADA Software from ETM (Siemens subsidiary), http://www.etm.at

[3] JCOP Framework, http://jcop.web.cern.ch

[4] UNICOS Framework, http://unicos.web.cern.ch

[5] Jira issue and project tracking software from Atlassian, https://www.atlassian.com/software/jira

[6] Ansible, https://www.ansible.com/

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV018

LINAC-200 GUN CONTROL SYSTEM: STATUS AND PLANS

M. Nozdrin*, V. Kobets, V. Minashkin, A. Trifonov
Joint Institute for Nuclear Research, Dubna, Russia

Abstract

Due to the development of the global Tango-based control system for Linac-200 accelerator, the new electron gun control system software was developed. Major gun electronics modification is foreseen. Current gun control system status and modification plans are reported.

INTRODUCTION

Linac-200 accelerator [1] is the core of the new electron test beam facility in JINR. It's quite hard to obtain necessary beam quality with standalone control subsystems, so the new global control system based on Tango is being developed [2]. In the frame of this work the existing gun control system part based on a standalone PC should be exchanged to a new one (Tango device & corresponding client).

HARDWARE

Electron Gun

Triode MIT-designed DC gun is used [3, 4]. Gun is placed in the SF_6 filled tank (pressure 6 atm) for better electric durability. Gun and cathode parameters are given in Tables 1 and 2 correspondingly.

Table 1: Parameters of the Linac-200 Gun

Type	Thermionic
Max. energy	400 keV
Beam intensity (peak)	200 mA
Normalized emittance	8π mm mrad
Pulse duration	200 mA

Table 2: Parameters of the Linac-200 Gun Cathode

Type	Dispenser
Max. energy	W + 20% Ba, Ca & Al oxides
Diameter	8 mm (S = 50 mm^2)
Working I & U range	6.5 V / 4 A to 8.8 V / 5 A
Lifetime	15000–20000 hours

Cathode is mounted at the end of the HV column — multisectional vacuum insulator separating the vacuumed beam chamber and the gas-filled gun tank. Gun electro-optical system consists of the extractor electrode and 15 anodes with forced resistive (R = 200 M) potential distribution (about 30 kV per gap).

First focusing anode voltage can vary in the range from 8 to 20 kV. Extractor electrode pulse is generated by Marx generator. Possible voltage is from −400 V (turnoff voltage) to 5 kV, pulse length—from 100 ns to 50 µs.

Schematic view of the Linac-200 electron gun control system structure is presented at Fig. 1. Controller board architecture is shown at Fig. 2. Gun optics and electronics scheme is shown at Fig. 3.

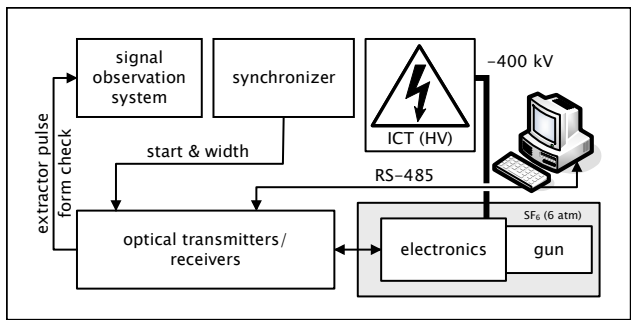


Figure 1: Linac-200 gun control system structure.

Gun Electronics

Cathode electronics is housed on the hot deck inside the tank. All cathode electronics assembly is "suspended" at a potential of −400 kV while the accelerator beamline is grounded.

Electronics consists of the following boards:

- Controller board acts as interface between the cathode electronics and the control computer.
- 50 kHz board supply transfiorms input voltage of 187 V / 50 Hz to 2 × 65 V / 50 kHz.
- Filament supply board sets the cathode filament current.
- Extractor pulser ensures the gun extraction pulse with necessary parameters.
- 1st focusing electrode voltage board is self-explanatory.

Controller Board

Board includes ATmega32 microcontroller, 4 DAC and 16 ADC channels, input register with 8 inputs, and output register with 8 outputs (TTL).

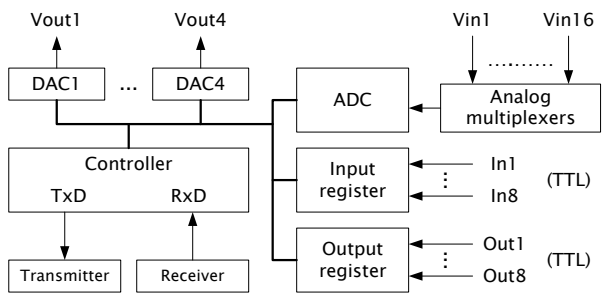


Figure 2: Controller board architecture.

* nozdrin@jinr.ru

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV018

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

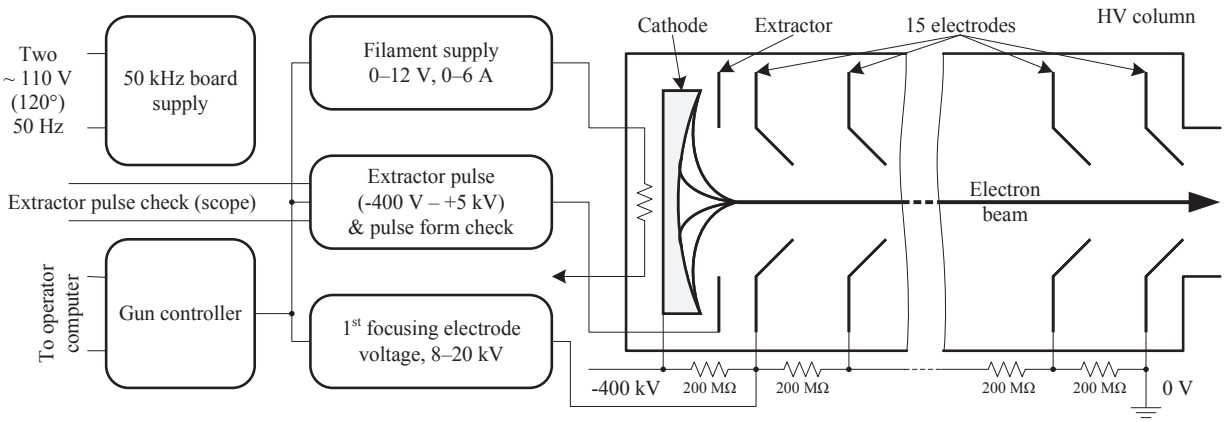


Figure 3: Gun optics and electronics.

Current implementation uses the following channels:

- 3 DAC are responsible for refetence voltages setting (to control filament supply, 1st focusing electrode and extractor voltages).
- 6 ADCs are used for measurement of the gun analog parameters (filament voltage and current, cathode electronics board voltage, pressure and temerature of the SF_6 gas, and 1st focusing electrode voltage).
- 2 input register stand for disrete parameters—status of two cathode electronics coolers in the tank (ON/OFF).

The rest channels are for future use / as spare ones.

Communication Line

To exchange information between the cathode electronics and outside equipment (−400 kV potential difference), optical fibers are used. Outside the gun RS-232 is used, than it converts to RS-485 to transmit information to control room (and receive it from it), and again to RS-232 to connect to the control PC.

New Setup

The only device changed in the frame of the integration of the gun control system to the global Tango-based system, is the control PC. It is changed to the same form-factor 4U industrial PC, which now runs under Debian 9. Tango server runs on this PC, client—at operator PC (as all the clients do). All the rest equipment, including the communication line, remains in service so far.

SOFTWARE

Communication Protocol

Information exchange between the gun controller and control computer is performed via the following protocol. PC (master) sends four bytes (device address, controller function to perform, and two data bytes) and waits an aswer from the controller. Delay between bytes is about 50 µs. Gun controller (slave) receives these four bytes, and sends an answer:

- Three bytes in the case of normal operation. First byte is the same as in control message, the other two contain data.
- Two bytes in the case of error. First byte again repeats the control message one, second byte contains an error code.

Timing diagram of one data exchange cycle is presented at Fig. 4. Full duration of one cycle is about 1 ms.

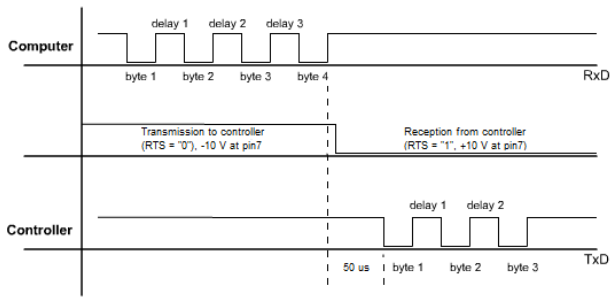


Figure 4: Timing diagram of one data exchange cycle between gun controller and control PC.

Standalone

The control software version developed for initial Linac-200 startup [5] was written in the Borland Pascal 7.0 under DOS and was interacting with the serial port using the `Port` predefined array. Next version was developed in Borland Delphi (Object Pascal) and was running under Windows XP [6].

Tango Device Class (Server)

Tango class `Gun` is written in C++ and runs under Debian 9. Due to communication protocol nonstandartness the UART driver was also included (instead of using corresponding communication device classes from Tango catalogue. Class structure in Pogo class generator is shown at Fig. 5.

Tango Device Client

The client application (Fig. 6) provides a graphical interface for controlling the electron gun. It is implemented in

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV018

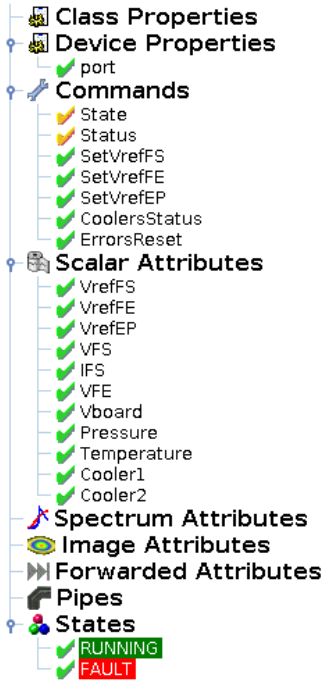


Figure 5: Gun class structure in Pogo.

C++ using the Qt5 framework and QTango library, and has the following functionality:

- control of the filament supply reference voltage;
- control of the 1st focusing electrode reference voltage;
- control of the extractor reference voltage;
- monitoring of the filament supply real voltage and current;
- monitoring of the 1st focusing electrode real voltage;
- provides information about the electron gun device server status.

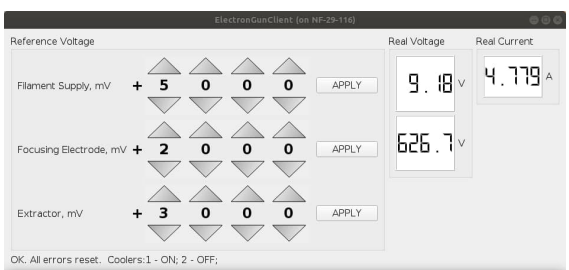


Figure 6: Gun client application.

Some gun parameters (SF_6 pressure and temperature, cathode electronics board voltage) are not presented, because corresponding sensors are not installed yet.

Emulator Tests

For preliminary testing the gun emulator was developed. The initial idea was to develop it also as Tango device by making use of an event mechanism. But it turned out that minimal polling interval is 5 ms (which is not sufficient). So the emulator was written as a common Linux C++ console application.

The emulator is a "semihardware" one, it takes one COM-port and acts like it is a gun controller I/O. So it's possible to work both with two PCs (one represents "gun", another—control computer), or with one PC representing both "gun" and control computer at different COM-ports.

The first configuration (two PCs) was used during emulator development and debugging. New PC with emulator was connected by RS-232 to the old gun operator PC. The emulator was considered as fully ready only when it could be fully recognized by the old (standalone) GunCtrl software as the real gun (no any errors, correct data exchange etc.)

After that, debugging of the Tango device class has been performed (now using one PC configuration with two ports connected by RS-232).

CONCLUSION & OUTLOOK

- C++ Tango device class and QTango client for Linac-200 electron gun were developed.
- C++ gun emulator was developed for testing.
- All software was successfully tested with emulator, tests with equipment will start after the end of the accelerator shutdown.
- The following hardware R&D is planned:
  - Bench for cathode electronics tests and repair works.
  - Full redesign of the cathode electronics using modern components.

REFERENCES

[1] M. A. Nozdrin *et al.*, "Linac-200: A new electron test beam facility at JINR", in *proc. 12th Int. Particle Accelerator Conf. (IPAC'21)*, (Campinas, Brazil), JACoW Publishing, May 2021, pp. 2697–2699. DOI: 10.18429/JACoW-IPAC2021-WEPAB042.

[2] A. Trifonov, M. Gostkin, V. Kobets, M. Nozdrin, A. Zhemchugov, and P. Zhuravlyov, "The control system of the linac-200 electron accelerator at JINR", in *proc. 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, (Shanghai, China), JACoW Publishing, Oct. 2021.

[3] J. Haimson, "A low emittance high duty factor injector linac", in *Proc. 4th Particle Accelerator Conf. (PAC'71)*, (Chicago, IL, USA), JACoW Publishing, Mar. 1971, pp. 592–595.

[4] F. B. Kroes *et al.*, "Improvement of the 400 kV linac electron source of AmPS", in *proc. 3rd European Particle Accelerator Conf.*, (Berlin, Germany), JACoW Publishing, Mar. 1992, pp. 1032–1034.

[5] N. I. Balalykin, V. P. Minashkin, M. A. Nozdrin, and G. D. Shirkov, "Control system of injector for linear electron accelerator LINAC-800", *Physics of Particles and Nuclei Letters*, vol. 7, no. 7, pp. 525–528, 2010. DOI: 10.1134/S1547477110070216.

[6] M. A. Nozdrin, N. Balalykin, V. Minashkin, V. Y. Schegolev, G. Shirkov, and G. V. Trubnikov, "Progress of the JINR e-Linac accelerator test-bench control systems", in *Proc. 9th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'12)*, (Kolkata, India), JACoW Publishing, Dec. 2012, pp. 203–205.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV019

PVEcho: DESIGN OF A Vista/EPICS BRIDGE FOR THE ISIS CONTROL SYSTEM TRANSITION

K.R.L. Baker*, I.D. Finch, G.D. Howells, A. Saoulis,
ISIS Neutron and Muon Source, Didcot, United Kingdom
M. Romanovschi, University of Manchester, United Kingdom

Abstract

The migration of the ISIS Accelerator Controls System from Vsystem to EPICS presents a significant challenge and risk to the day-to-day operations of the accelerator. An evaluation of potential options has indicated that the most effective migration method to mitigate against this risk is to make use of a 'hybrid' system running Vsystem and EPICS simultaneously. This allows for a phased porting of controls hardware from the existing software to EPICS. This work will outline the prototype Vsystem/EPICS bridge that will facilitate this hybrid operation, referred to as PVEcho. The bridge has been developed in Python, utilising existing communication from Vsystem to an MQTT broker developed as part of a previous project. Docker containers have been used for its development to create a test environment that allows the software to communicate with other active services currently used at ISIS.

INTRODUCTION

The ISIS Neutron and Muon Source [1] has been operated using Vista Control Systems' commercial product Vsystem [2], colloquially referred to as Vista, since 1998. Recently, a study was undertaken to evaluate its use against that of the Experimental Physics and Industrial Control System [3] (EPICS) at ISIS. The study determined that the control system should be migrated to EPICS [4]. One of the key benefits of the migration will be to promote collaboration with other facilities at Rutherford Appleton Laboratory, such as ISIS Instrumentation [5], Central Laser Facility [6] and Diamond Light Source [7], also using EPICS.

In order to minimise the impact on business-as-usual at the facility, a phased porting of control of hardware is the desired option for the transition. This requires the development of bridging software that will map approximately 33,000 channels that exist in Vsystem to an equivalent EPICS Process Variable (PV). This will allow a progressive migration of the operator control screens in the main control room (MCR) alongside a gradual transition of hardware without interrupting operator control. The current control screens are produced using Vsystem's Vdraw [8] tool but will be migrated to screens created using CS-Studio Phoebus [9] as part of the transition. Phoebus is the suite of applications that can be used to display screens as well as probe EPICS PVs.

The use of a bridge also elegantly permits some legacy hardware that cannot be converted to EPICS to still be run using Vsystem but using Phoebus control screens. PVEcho will serve as this bridge, responsible for replicating the behaviour of the current control system in EPICS.

PVEcho

Each component of hardware that comprises the accelerator controls system is associated with a group of unique channels through which value changes in the hardware can be communicated. During the migration, we will need to exactly replicate each of these channels that currently operate through Vsystem as an equivalent EPICS PV, including their value, alarm and display configuration. Each Vsystem channel and corresponding mirrored EPICS PV needs to be kept synchronised. This in turn will allow the control system to operate with either the Vsystem channel or the EPICS PV acting as the source of truth.

In the early stages of the transition, the majority of channels will still be operated through Vsystem while long-term EPICS equivalents are being developed. In this case, the source of truth will be the Vista channels and changes in their values and alarm limits will be broadcast from Vista to EPICS, handled by the vistatoepics (v2e) program. Once the control of hardware of specific channels has been ported to EPICS, the source of truth will transfer. Now, changes to the PV will be monitored and transferred from EPICS to Vista via MQTT [10] messages. This is handled by the epicstovista (e2v) program. The vistatoepics and epicstovista applications work in tandem to make up the PVEcho bridge, a visual representation of which can be seen in Fig. 1. The long-form names are used for descriptions of the applications, while the shorthand versions (v2e and e2v) are used for simplicity in the codebase and in diagrams. More details about the individual programs will be given in subsequent sections of this paper.

To track which channels have Vista or EPICS representing their true value, we have created a `PVMonitor` utility class to use in both programs. A list of channels to monitor are defined in a file which the `PVMonitor` regularly checks for changes (in our case once per second). When a change is noted, the monitor determines which channels have been added or removed relative to the file's previous state, and starts or stops tracking those channels accordingly. This method allows a lot of control and flexibility over the channels to be replicated in EPICS, allowing the transition of individual channels at a time. As the same class is used in both applications, "channel" may be substituted for PVs when referring to epicstovista, where the same approach is used.

* k.baker@stfc.ac.uk

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV019

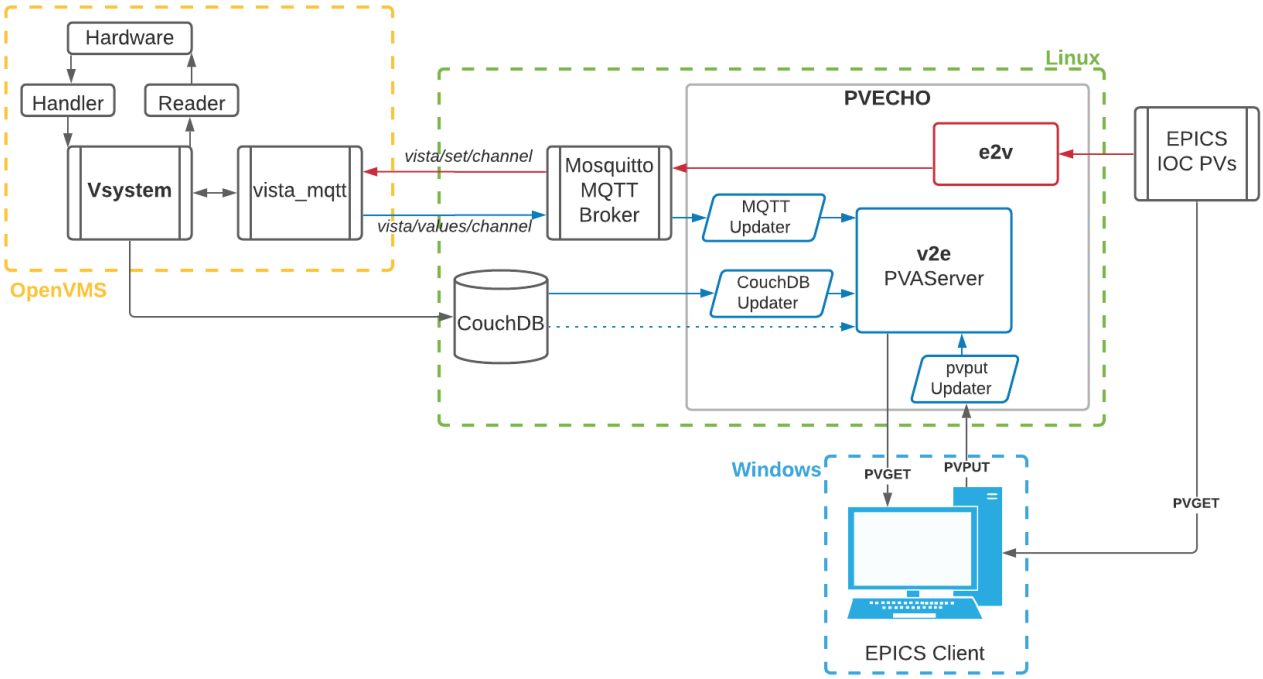


Figure 1: An overview of how PVEcho will integrate into the ISIS Accelerator Control System. VSystem will remain operating on OpenVMS, while newer systems and EPICS will operate on Linux, with the MCR running Phoebus [9] screens through Windows. Arrows in dark blue represent the flow of data from a Vsystem channel through to EPICS (v2e). They also indicate the Updater classes that parse incoming data. Dotted lines indicate an initialization step in the creation of PVs. Arrows in red follow the flow of information from EPICS to Vsystem (e2v). PVEcho interfaces to Vsystem through a Mosquitto MQTT broker [10] and the vista_mqtt service [11].

Both vistatoepics and epicstovista components of PVEcho will leverage the MQTT messaging service linked to Vista, named vista_mqtt, set up as part of a previous project within the ISIS controls group [11]. The vista_mqtt service interfaces to Vsystem through topics associated with each Vsystem channel. When a value in Vista changes, the application (originally developed in Python but being migrated to C++) generates an MQTT message with the updated value and sends it to a read-only "values" topic associated with the channel. If a message is published to the channel's "set" topic, the application triggers a Vsystem set command to change the value of the channel within the control system. Vistatoepics will make use of the former, read-only case, while epicstovista will utilise the writable topics.

By subscribing to the MQTT value topic associated with a Vsystem channel we can monitor changes to its value through the clients and callbacks available in MQTT's python library [12]. The changes are then echoed to EPICS through vistatoepics. Similarly, we track changes to the EPICS PVs through `pvapy`'s [13] `Channel` client and we set the value of its corresponding channel in epicstovista by publishing a message to the channel's set topic in MQTT.

The programs will also exploit the CouchDB [14] database set up by the group to store metadata associated with each Vsystem channel. The database stores information such as the channel type (e.g. integer, float, boolean, string) and display formatting, as well as alarm type and limits if applicable. In the case of vistatoepics, this information is used to define the structure of the PV to be created. In both programs it is also used to define how to parse incoming values to a format appropriate for the system where it will be echoed, based on the channel's type.

VISTA TO EPICS (V2E)

The PVEcho bridge is designed to replicate the entire Vista control system, rather than individual devices that usually constitute a distributed control system. This places a focus on the ability to dynamically add, remove or edit PVs on the server created to host the replica PVs without having to restart the program, which would interrupt operation of the accelerator. Therefore, a PVAServer from the `pvapy` library was favoured over the traditional IOC for the vistatoepics component of PVEcho. On running the vistatoepics program, the configuration that dictates the services to connect to and the necessary files to be read into memory is loaded and parsed. In this case, files include a record of the CouchDB entries containing the metadata for the Vsystem channels, the mappings of reference channels to their setter channels and the list of PVs to monitor. The benefit of using this approach is that we can tailor our configuration file for production or development and testing purposes.

Once all of the relevant classes have been instantiated, we enter the main loop of the program. It continuously checks

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the monitor file for changes, acts accordingly for the PVs that have been added or removed from the list and then goes on to check for any changes to CouchDB. This continuously runs until the program is interrupted, as illustrated by Fig. 2.

Initialisation

On start-up and initialization of the vistatoepics component of PVEcho, a snapshot of the existing configuration for each of the channels on Vsystem is captured using the ISIS CouchDB database. The channel metadata (such as the display and alarm configuration) associated with each channel in the monitor list is then translated into a format appropriate for EPICS.

Monitoring

Once the PVs have been created on the server, changes to their value and metadata are tracked through three different avenues: MQTT topic messages from Vista, CouchDB updates and EPICS `pvput` commands. Each source produces updates in a different `json` format so a fleet of customised 'Updater' classes are used to manipulate the data from the raw input into a structure appropriate for use with the `pvapy PvObject.set()` method which is used to apply the changes to the PV. The PV is then updated on the server so that changes are visible to EPICS clients such as Phoebus [9] control screens.

1. **MQTT**
   As described previously, once a PV has been created, we subscribe to the original channel's corresponding MQTT topic, published from Vsystem, which tracks changes to the PV's value.

2. **CouchDB**
   CouchDB shadows the Vsystem databases for each channel so it acts as a source of truth for the channel metadata that is not accessible through MQTT. To keep PVs consistent with their channel, we track changes to Vsystem database files through alterations to CouchDB using the CouchDB Python client [15] and modify the PV's metadata to match the updated values from Vista.

3. **pvput**
   PV values and metadata may also be changed by the operators through `pvput` commands using Phoebus control screens. As we are using custom classes that inherit from the original `pvapy PvObject` class for different channel types, when an update comes through we need to ensure that our object is updated correctly on the server.

Alarm Management

The vistatoepics component of PVEcho is also responsible for mirroring the current alarm management in Vsystem. Implementing the alarms from the outset will allow us to start using EPICS alarm handling and monitoring software throughout the transition, as well as minimise operational changes for the crew. The mapping of range and binary match alarms is relatively straightforward as EPICS defines these alarm types natively. Vsystem also offers the option of integer match and reference alarms [16], which do not have a corresponding native type in EPICS.

An example of where an integer match alarm might exist is a pump that operates in multiple modes. These could include pump operating states such as `0=off, on=1, 2=fault, 3=standby, 4=turbo`, where only state 2 should trigger an alarm.

A reference alarm is a channel where the alarm limits are determined by the value of a different Vsystem channel according to some calculation. The channel that dictates the reference channel's alarm limits is referred to as its "setter" channel. The alarm limits are recalculated for the reference channel whenever the value of the setter channel changes.

In order to replicate Vsystem as closely as possible within EPICS, the logic behind the Vista alarms has been implemented manually within the `Updater` classes. This was primarily possible because of the freedom of control over alarms that the `pvapy` library allows as well as the availability of the alarm definitions (for example type, limits, setter channel) for each channel in CouchDB.

- **Range** With each new update to the value or the metadata, the current value is compared to the most recent alarm limits to determine its alarm state, either MAJOR, MINOR or NO ALARM.
- **Binary Match** The alarm state is set to raise a major alarm if the current value matches that of the defined alarm match value.
- **Integer Match** The alarm state for integer match alarms is determined by comparing the value of the PV with its highAlarmLimit. On initialisation, this and all the other limits are set to be equal to the alarm match value from Vsystem. To circumvent the issue where a user could change the value of the match alarm through one of the other limits (e.g. lowAlarmLimit), a custom function has been devised and implemented in the `PvPutUpdater` (seen in Fig. 1), whereby if the value of any of the alarm limits is changed on a match alarm PV, all of the alarm limits are modified to equal the updated value. We use the same approach to update the severity of the match alarm.
- **Reference** A mapping of reference alarm PVs and their setter channels are stored in memory. Whenever a PV updates, a check is performed to see whether the channel is a setter. If so, all of its reference channels are also updated to reflect the new limits and their alarm state is recalculated with the new limits.

EPICS TO VISTA (E2V)

As previously mentioned, currently the ISIS control screens are created and run using Vista's VDraw tool [8]. The intention during the transition is to operate a hybrid User Interface (UI), transferring some screens to Phoebus/EPICS while others remain in VDraw/VSystem as control of hardware is ported. If a user changes the value of the PV in

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV019

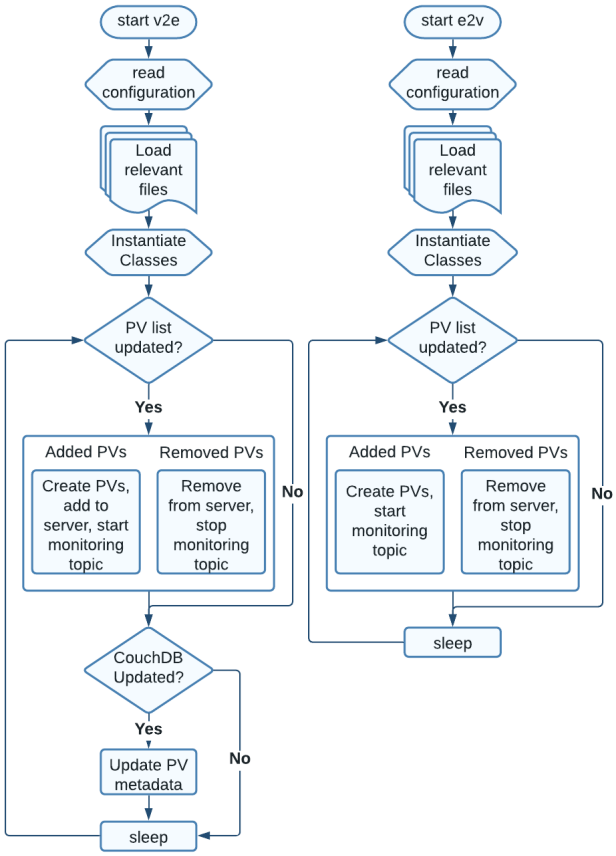


Figure 2: Flowcharts demonstrating the logic behind vistatoepics and epicstovista.

the ISIS MCR through a Phoebus screen where the hardware is still controlled by Vsystem, the change needs to be propagated from the EPICS Client to Vsystem to update the hardware. The same is true in reverse; if hardware is read via an EPICS IOC but the control screen is still operated by a VDraw screen, the PV values will need to be propagated to Vsystem. A `pvapy Channel` connection and callback is used to do this. As the update comes through via the EPICS callback, the values are converted to a JSON format that the Vsystem code on OpenVMS is expecting. The message is then published to the correct setter topic on the MQTT broker, which activates a set command in Vsystem [11]. The main loop of the program follows a similar structure to vistatoepics and can be seen in Fig. 2.

Alarmed channels are monitored in a similar way, but changes to the channel metadata are also tracked to provide updates if the alarm limits are changed through EPICS.

DEVELOPMENT ENVIRONMENT

Environment

The ISIS accelerator control system is also transitioning from OpenVMS to Linux as a base operating system. Therefore, in order to future-proof the design of the bridging software, PVEcho has been developed using Docker [17] environments with a Linux operating system as the base of the containers, to isolate and replicate PVEcho's operating environment. Two separate containers will be used to run vistatoepics and epicstovista simultaneously, while isolating the processes to give greater control over their execution.

Services

Alongside Linux and EPICS, PVEcho requires a connection to a number of other active services that ISIS utilises to monitor and maintain its control system. These include the ISIS MQTT broker [11] and the instance of CouchDB where channel metadata is stored. The use of Docker while developing PVEcho allowed connection to personally managed containers to prevent premature interaction with the live control system, minimising the risk of unintentional changes to existing channels.

Testing

Throughout the development of the software, we have tried to implement software good practice wherever possible. This includes the application of unit and integration testing utilising GitLab's CI/CD facility and the unittest [18] framework in Python. Performance has also been monitored using an in-house virtual logging system [19] to track the CPU and memory use of each container while the software is running.

FUTURE WORK

While functional, work on the current implementation of PVEcho is still ongoing. The accelerators at ISIS will primarily use the newer pvaccess protocol rather than channel access [11]. The current version of PVEcho only implements PVs using the pvaccess protocol, as highlighted by the use of a PVAServer to host the PVs. Future work on the software will incorporate channel access, ensuring that any changes to Vsystem channel values are broadcast to channel access PVs as well as their pvaccess equivalents.

Future developments will also include the addition of delayed alarms that are currently in use with VSystem. Additionally, the PVMonitor class should be modified to allow the bulk addition and removal of PVs from the monitor list as devices containing groups of PVs are migrated, to reduce the risk of user error in the transfer of channel names. Stress testing of the live system should also be completed to ensure that the software can cope with the day-to-day frequency of messages and surges in demand.

CONCLUSION

A prototype of the bridge connecting Vsystem to EPICS has been created to mimic more than 33,000 channels live on the ISIS accelerator control system. PVEcho will allow a hybrid system running both Vsystem and EPICS to be utilised as control of hardware is progressively migrated to EPICS, without incurring the risk of interrupting accelerator operations. The prototype is due to be tested in deployment before the end of the 2021 ISIS long shutdown [20].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] J.W.G. Thomason, "The ISIS Spallation Neutron and Muon Source—The first thirty-three years", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 91, pp. 61–67, Feb 2019. doi:10.1016/j.nima.2018.11.129

[2] Vista Controls Systems Inc., https://www.vista-control.com/.

[3] EPICS Control System, https://epics-controls.org/.

[4] I.D. Finch, "Evaluating VISTA and EPICS With Regard to Future Control Systems Development at ISIS", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 291–292. doi:10.18429/JACoW-ICALEPCS2019-MOPHA042

[5] K.V.L. Baker, F.A. Akeroyd, J.R. Holt, D.P. Keymer, T. Löhnert, C. Moreton-Smith, *et al.*, "IBEX: Beamline Control at ISIS Pulsed Neutron and Muon Source", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 59–64. doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

[6] Central Laser Facility, Rutherford Appleton Laboratory, United Kingdom, https://www.clf.stfc.ac.uk

[7] Diamond Light Source, Rutherford Appleton Laboratory, United Kingdom, https://www.diamond.ac.uk

[8] Vdraw, Vsystem User's Guide v4.3, Vista Controls Systems Inc., Jan 2014

[9] CS Studio Phoebus, https://controlssoftware.sns.ornl.gov/css_phoebus/.

[10] MQTT standard: MQTT Version 3.1.1, OASIS, October 2014, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[11] I.D. Finch, A. Saoulis and G.D. Howells, "Controls Data Archiving at the ISIS Neutron and Muon Source for in-depth analysis and ML applications", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, paper WEPV049, this conference.

[12] Eclipse Paho MQTT Python client library, https://pypi.org/project/paho-mqtt/.

[13] pvapy, https://epics.anl.gov/extensions/pvapy/production/index.html

[14] Apache CouchDB, http://couchdb.apache.org/.

[15] CouchDB's python client, https://couchdb-python.readthedocs.io/en/latest/.

[16] Vaccess Concepts, Vsystem User's Guide v4.3, Vista Controls Systems Inc., Jan 2014.

[17] Docker, https://www.docker.com/.

[18] python unittest framework, https://docs.python.org/3/library/unittest.html

[19] G.D. Howells and I.D. Finch, "Containerised Control Systems Development at Isis and Potential Use in an Epics System", in *presented at ICALEPCS'21*, Shanghai, China, Oct. 2021, paper WEPV050, this conference.

[20] ISIS Long Shutdown 2021 Update, https://www.isis.stfc.ac.uk/Pages/ShutdownUpdate.aspx.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV020

DIGITISATION OF THE ANALOGUE WAVEFORM SYSTEM AT ISIS

W. A. Frank*, B. R. Aljamal†, R. Washington
STFC Rutherford Appleton Laboratory, Didcot, Oxfordshire, UK

Abstract

The Analogue Waveform System (AWS) at the ISIS Neutron and Muon Source is a distributed system that allows operators to select and monitor analogue waveforms from equipment throughout the facility on oscilloscopes in the Main Control Room (MCR). These signals originate from key accelerator systems in the linear accelerator and synchrotron such as the ion source, magnets, beam diagnostics, and radio frequency (RF) systems. Historical data for ISIS is available on the control system for many relevant channels. However, at present, to avoid disrupting the oscilloscope displays in the MCR, only an hourly image capture of the AWS waveforms is stored. This is largely inadequate for potential data-intensive applications such as anomaly detection, predictive maintenance, post-mortem analysis, or (semi-)automated machine setup, optimization, and control. To address this, a new digital data acquisition (DAQ) system is under development based on the principle of large channel count, simultaneous DAQ. This paper details the proposed architecture of the system and the results of initial prototyping, testing, and commissioning.

INTRODUCTION

The Analogue Waveform System has been a key part of accelerator operations since the inception of the ISIS Neutron and Muon Source in 1984 [1]. At present, it is a distributed analogue waveform switching system that allows operators to select and monitor signals from equipment throughout the facility from the Main Control Room (MCR).

Recently the particle accelerator community has been turning to the increasingly mature fields of machine learning and data science to help manage the inherent complexity of these facilities [2]. Therefore, to modernise the current system in order to meet the data needs of these techniques, a new digital data acquisition system (DAQ), called the Digitised Waveform System (DWS), is under development. The new system aims to increase the signal capacity, improve flexibility, as well as signal integrity of the system. Further benefits are expected from the ability to store and archive descriptive statistics of the waveforms and raw waveforms for specific applications.

EXISTING AWS SYSTEM

The AWS consists of three main cable trunks. Each trunk can, in principle, connect any number of ISIS Controls CPCI System (CPS) crates [3] using a daisy chain topology. Each CPS crate provides 32 analogue inputs to the system. However, at most six signals from all the CPS crates connected on a single trunk can be multiplexed to a bank of Tektronix TDS3014B oscilloscopes in the MCR.

The CPS crates contain a Kontron CP3004-SA CPU board, Pickering PXI switching matrix modules (40-531-022/40-531-023), a custom six channel fully differential amplifier board, and a seven segment I/O display board. Inter-crate communication is over the CompactPCI [4] backplane. The direct analogue trunks provide negligible signal transmission latency but limit the flexibility of the system and make expanding the signal capacity difficult. The fast update rate of the oscilloscopes ensure real-time capture of each 50 Hz acceleration cycle.

There are currently 13 AWS CPS crates installed across the facility, providing 416 inputs to the system. Control and status monitoring of the AWS as well as triggering from the ISIS Timing System is provided by the ISIS Control System [5], based on the VSystem SCADA/HMI toolbox from Vista Control Systems [6] .

DWS SYSTEM OVERVIEW

Hardware

Recent technological developments have brought down the cost of high-performance and high channel count digitisers. This has made possible the option to replace the multiplexed analogue approach with a largely digital paradigm consisting of network connected digitisers. A similar one-digitiser-per-device approach with the DAQ placed as close to the accelerator device as possible has recently been proposed and implemented at the Facility for Antiproton and Ion Research (FAIR) [7].

For the DWS, the selected DAQ hardware is the ACQ2106 carrier units from D-TACQ Solutions fitted with two or three ACQ482 ELF modules [8]. These digitisers provide excellent analogue and digital performance at a competitive cost per channel. In addition, it has built-in EPICS support, providing extensive monitoring, supervisory and control features that reduces integration efforts for the new EPICS based control system at ISIS [9]. Each ACQ482 has 16 differential inputs with an analogue bandwidth of 10 MHz and ADC clock speeds up to 80 MSPS. The bandwidth is sufficient to digitise the ≤1 MHz analogue bandwidth of the existing AWS. The signals from the AWS are connected to the VHDCI input connectors of the ACQ482 with 32 channel dual pin LEMO 1U, 19" panels. A total of nine ACQ2106 units will cover the existing system capacity of 416 channels, with the option to install additional units to cover any future capacity requirements.

The Rear Transition Module (RTM-T) in the ACQ2106 units connects to the DAQ server fitted with AFHBA404 PCI Express Host Bus Adapters via LC terminated OM3

* william.frank@stfc.ac.uk
† basil.aljamal@stfc.ac.uk

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

50/125µm optical fiber. Avago AFBR-709SMZ SFP+ transceiver modules are used for the ACQ2106 and AFHBA404 connections. The AFHBA404 is a standard PCIe x4 Gen 2.0 card which provides four SFP+ transceiver ports at up to 6 Gbps [10]. The DAQ server is a Dell PowerEdge T440 Tower Server configured to be rackmountable fitted with an Intel Xeon Gold 5120 CPU at 2.20 GHz. The server has 256 GB of RAM installed and is fitted with a 3.5 TB SSD. A demonstration system, shown in Fig. 1, has been setup and tested for use with signals from various diagnostics systems at ISIS.

Figure 1: The demonstration system for the DWS connected to signals from various diagnostics systems at ISIS.

The DAQ is triggered and synchronised with the ISIS Timing System (200 kHz timing signals over twisted pair (RS-485)) via in-house CPS Type-T timing cards. An in-house 1U, 19" Timing Trigger Mixer unit is used to convert the differential RS-485 signals on the CPS Type-T BNO connectors to TTL compatible single-ended pulses for use with the external single-pole LEMO trigger input on the ACQ2106s. An overview of the DWS is shown in Fig. 2 and a summary of the specifications for hardware components of the DWS system is shown in Table 1.

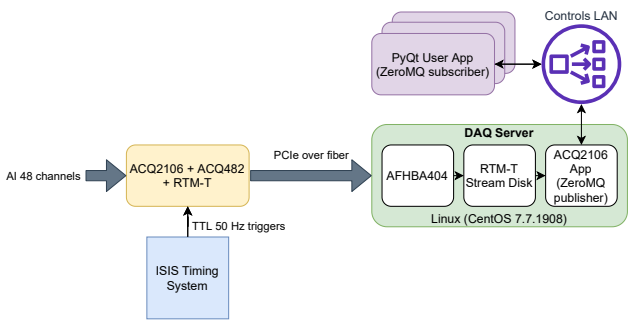


Figure 2: Overview of the DWS.

Software

The DAQ server is set up with CentOS Linux release 7.7.1908 (Core) patched with the real-time Linux kernel.

Table 1: DWS Hardware Specifications

Item	Specification
Dell PowerEdge T440 Tower Server	Intel Xeon Gold 5120 CPU at 2.20 GHz, 14 cores, RAM 256 GB, SSD 3.5 TB.
D-TACQ Solutions ACQ2106, x3 ACQ482, MGT482	48 simultaneously sampled differential analogue inputs, 14-bit, 1 MS/s, ±10 V. PCI-Express over fiber, streaming data over 400 MB/s.
D-TACQ Solutions LEMO Panel	1U 19" rack-mount panel. 32 dual pin LEMO connectors. Differential connection. 100 V Zener diode transient protection.
D-TACQ Solutions AFHBA404	PCIe x4 Gen 2.0 card which provides four SFP+ transceiver ports at up to 6 Gbps.
CPS Type-T, Timing Trigger Mixer	2.5 µs TTL compatible timing trigger pulses at 50 Hz on single-pole LEMO. Remotely configurable delay time with a resolution of 5 µs.

The AFHBA404 Linux device driver is used along side the RTM-T-stream-disk application software [11], both released by D-TACQ Solutions under GNU General Public License v2.0 [12]. The AFHBA404 writes 2 MB buffers directly to memory using DMA operations. Each 2 MB buffer contains 16384 samples, on 48 channels, 16-bit data which corresponds to 16 ms of data on each channel at 1 MS/s. The driver provides the address for the DMA engine to do this. RTM-T-stream-disk creates a series of fixed size binary files in RAM which are overwritten cyclically. The binary data files are consumed by a custom, in-house developed, application that employs the Linux inotify API [13] to monitor the data files and return events when the data files have been written and closed. The data files are then read out in real-time, before they are overwritten by the driver. The ZeroMQ protocol is used for data exchange between the DAQ server and other client applications over controls network using the publish/subscribe messaging pattern to achieve minimum latency. Latency measurements have been performed on the ISIS machine controls network which showed that latency is linearly related to message size. An average latency of 2–6 ms was found for expected message sizes of 30–60 kB. Throughput measurements were also performed over varying message sizes which showed negligible impact on throughput with an approximate throughput of 94 Mbit/s. ZeroMQ was selected as it allowed greater flexibility in terms of client host operating system and programming language, as well as high scalability across multiple machines. Every channel of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV020

data uses a separate TCP port and is identified with a unique ZeroMQ topic [14]. The data stream is currently started and stopped remotely using the acq400 Host API Python module provided by D-TACQ. The plan is for this to be migrated into the user application in future.

For data visualisation, use of EPICS based Phoebus user displays was investigated. However, it was found that the update rate of approximately 3 Hz was not sufficient for the real-time waveform requirements of the system (4 channels, 16-bit data, 16384 points per waveform at 50 Hz). As a result, a PyQt based application has been developed for real-time user application data visualisation called "DWS Live Viewer" (Fig. 3). The application is a ZeroMQ client that subscribes to selected channels streams and plots them in real-time.

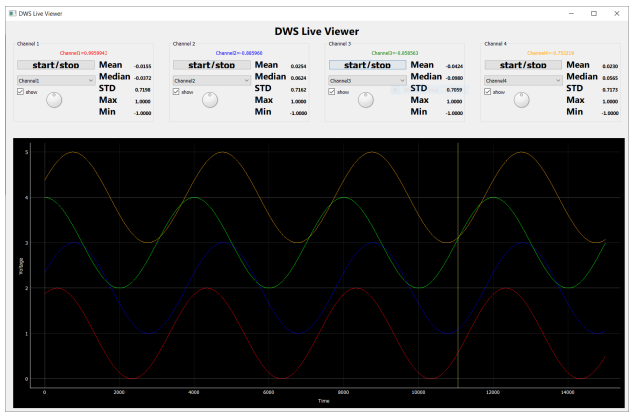


Figure 3: DWS live viewer.

Data Storage Strategy

One of the most important features in the new DWS system is that signals are available in a digital form so digitized waveform storage becomes possible.

An approximate calculation was done to estimate the required storage capacity for 416 signals sampled at 1 MS/s for 16 ms interval synchronised with the machine repetition clock (50 Hz) for 10 weeks run cycle which is around 3.8 petabytes. This volume of data is beyond our capacity to maintain, and therefore the following short-term data storage strategy will be used:

A one minute long ring buffer of raw data for each signal used for diagnostics purposes especially during machine startup and beam commissioning. Another use case is post-mortem data logging. In this case, when an interlock occurs pre-interlock data writing stops and post-interlock data writing starts for a specified amount of time.

For long-term data storage, the proposed solution is to store statistical properties (mean, median, standard deviation, minimum, and maximum) of the raw data rather than storing the waveform data itself.

Raw data from each channel will be received by subscribing to the ZeroMQ stream with the corresponding topic for the channel of interest. Statistical properties will then be calculated and pushed into Telegraf which will aggregate data points into big chunks for efficient data writing into InfluxDB [15]. InfluxDB is a high-performance time-series database optimised for query and visualization of timestamped data . With this solution, a significant reduction in storage capacity and pre-processing overhead can be achieved.

Some applications may require raw waveform data because statistical properties alone aren't sufficient. For cases like this, a ZeroMQ based stream processing pipeline will be implemented to do online data processing rather than batch processing on big data sets of stored raw data.

A summary of the software components of the DWS system is shown in Table 2.

Table 2: DWS Software Specifications

Software	Version	Reference
CentOS Linux	7.9.2009	https://www.centos.org/
AFHBA404 Linux device driver	v2.3	https://github.com/D-TACQ/AFHBA404/
RTM-T-stream-disk	2.3	https://github.com/D-TACQ/AFHBA404/tree/master/STREAM
inotify	Linux kernel ≥2.6.13, glibc ≥2.4	https://www.man7.org/linux/man-pages/man7/inotify.7.html
cppzmq	4.8.0	https://github.com/zeromq/cppzmq
acq400-hapi	2.8.2	https://github.com/D-TACQ/acq400_hapi
PyQt5	5.15.4	https://riverbankcomputing.com/software/pyqt/
pyzmq	22.1.0	https://github.com/zeromq/pyzmq
pyqtgraph	0.11.1	https://www.pyqtgraph.org/
InfluxDB	2.0.8	https://www.influxdata.com/products/influxdb
Telegraf	1.20.1	https://www.influxdata.com/time-series-platform/telegraf

SUMMARY AND FUTURE WORK

The proposed architecture for a new digital DAQ system for the digitisation, visualisation and storage of waveform

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

signals from the AWS at ISIS has been presented and described. A demonstration system has been setup and tested for use with signals from various diagnostics systems at ISIS. This demonstration system has been newly setup during a long shutdown in ISIS operation. As such, an Agilent 33220A function generator connected to the controls network is currently being used to provide representative waveforms for initial testing and commissioning. Remote control and configuration of the function generator is provided by the PyVISA Python package along with the Keysight IO Library Suite. Waveforms from the existing diagnostics systems will be targeted during the initial beam commissioning prior to the user cycle in January 2022.

Investigations are also underway for two additional tiers of hardware for both low bandwidth signals (< 100 Hz) and those above the bandwidth of the existing AWS (1–500 MHz) that currently sit outside the system. For slow signals a bespoke solution is under development. For the high speed signals, for example from the synchrotron extraction kicker, dedicated digitisers (< 10 channels total) based on MicroTCA [16] will be investigated.

Currently time-stamping of each cycle of data (at 20 ms intervals) is provided on the DAQ server which is synchronised with a NTP server. It is noted that the D-TACQ ACQ2106 digitiser units are White Rabbit (IEEE 1588-2019) [17] compatible and accurate sub-nanosecond time-stamping of the data could be investigated in conjunction with any future development of the ISIS timing system.

The ACQ2106 units are also supplied with calibration data. The calibration data allows client applications to convert raw binary data from the ADC to calibrated voltage readings. So far this has not been implemented but in future the data will be calibrated prior to visualisation.

ACKNOWLEDGEMENTS

The authors express their sincere thanks to Peter Milne at D-TACQ Solutions for his help and support with the initial prototyping, setup and testing of the D-TACQ Solutions equipment.

REFERENCES

[1] J. W. G. Thomason, "The ISIS Spallation Neutron and Muon Source—The first thirty-three years", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 917, pp. 61–67, February 2019. doi:10.1016/j.nima.2018.11.129

[2] A. Edelen, C. Mayes, D. Bowring, D. Ratner, A. Adelmann, R. Ischebeck, J. Snuverink, I. Agapov, R. Kammering, J. Edelen, *et al.*, "Opportunities in Machine Learning for Particle Accelerators", 2018. arXiv:1811.03172

[3] R. A. Washington, "Migrating to Tiny Core Linux in a Control System", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 920–922. doi:10.18429/JACoW-ICALEPCS2019-WECPL05

[4] CompactPCI Overview. https://www.picmg.org/openstandards/compactpci/

[5] B. Mannix, "Vista Controls' VSystem at the ISIS Pulsed Neutron Facility", presented at ICALEPCS'07, Knoxville, TN, USA, Oct. 2007, unpublished.

[6] Vista Control Systems, Inc. http://www.vista-control.com

[7] R. J. Steinhagen *et al.*, "Generic Digitization of Analog Signals at FAIR – First Prototype Results at GSI", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 2514–2517. doi:10.18429/JACoW-IPAC2019-WEPGW021

[8] D-TACQ 4G User Guide, D-TACQ Solutions Ltd, 2021. https://www.d-tacq.com/resources/d-tacq-4G-acq4xx-UserGuide-r36.pdf

[9] I. D. Finch, "Evaluating VISTA and EPICS With Regard to Future Control Systems Development at ISIS", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 291–292. doi:10.18429/JACoW-ICALEPCS2019-MOPHA042

[10] AFHBA404 Host Bus Adapter Product Specification, D-TACQ Solutions Ltd, 2016. https://www.d-tacq.com/acq400ds/afhba404-product-specification.pdf

[11] RTM-T User Guide, D-TACQ Solutions Ltd, 2012. https://www.d-tacq.com/pdfs/RTM-T-user-guide-v7.pdf

[12] GNU General Public License v2.0 https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

[13] inotify - Linux manual page. https://man7.org/linux/man-pages/man7/inotify.7.html

[14] ZeroMQ Socket API. https://zeromq.org/socket-api

[15] I. Finch, G. Howells, and A. Saoulis, "Controls Data Archiving at the ISIS Neutron and Muon Source for in-Depth Analysis and ML Applications", presented at the ICALEPCS'21, Shanghai, CN, Oct. 2021, paper WEPV049, this conference.

[16] MicroTCA Overview. https://www.picmg.org/openstandards/microtca/.

[17] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White Rabbit: Sub-nanosecond Timing Distribution over Ethernet", in *Proc. of 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Brescia, Italy, Oct. 2009, pp. 58–62.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

UPGRADING THE NATIONAL IGNITION FACILITY'S (NIF) INTEGRATED COMPUTER CONTROL SYSTEM TO SUPPORT OPTICAL THOMPSON SCATTERING (OTS) DIAGNOSTIC

A. Barnes, A. Awwal, L. Beaulac, B. Blackwell, G. Brunton, K. Burns, J. Castro Morales, M. Fedorov, R. Lacuata, R. Leach, D. Mathisen, V. Miller Kamm, S. Muralidhar, V. Pacheu, Y. Pan, S. Patankar, B. P. Patel, M. Paul, R. Rozenshteyn, R. Sanchez, S. Sauter, M. Taranowski, D. Tucker, K. C. Wilhelmsen, B. Wilson, H. Zhang
Lawrence Livermore National Laboratory, Livermore, USA

Abstract

With the ability to deliver 2.1 MJ of 500 TW ultraviolet laser light to a target, the National Ignition Facility (NIF) is the world's most energetic laser. This combination of energy and power allows the study of materials under conditions similar to the center of the sun. On fusion ignition experiments, plasma generated in the interior of the target shell can detrimentally impact the implosion symmetry and the resulting energy output. We are in the final stages of commissioning a significant new diagnostic system that will allow us to better understand the plasma conditions and improve our symmetry control techniques. This Optical Thompson Scattering (OTS) system consists of two major components: a probe laser beamline capable of delivering a world first 1 J of energy at 211 nm, and a diagnostic that both reflects the probe laser into the target and collects the scattered photons. Between these two components, the control system enhancements required integration of over 450 components into the existing automation suite. This talk will provide an overview of the system upgrade approach and the tools used to efficiently manage and test changes to both our data and software.

BACKGROUND INFORMATION

The purpose of the NIF is to continue research into nuclear fusion, specifically laser driven inertial confinement fusion (ICF). To accomplish this, NIF uses 192 lasers. When combined, the lasers can deliver up to 2.1 MJ of energy at 351 nm. For a sense of scale, the NIF building containing the lasers stands four stories tall and can fit three American football fields on the roof.

Responsibility for driving the NIF through its experiments lies with the Integrated Computer Control System (ICCS). At the lowest level, ICCS provides direct control of devices for manual operations during maintenance and troubleshooting tasks. On top of that, it provides multiple layers of automation which allow less than a dozen operators to configure and control more than 66,000 control points on over 2,300 processors and embedded controllers through the course of a shot cycle.

Behind the scenes ICCS has chosen a data driven architecture to keep things manageable. A predominantly Java code base of over 3 million lines of code provides the basic implementation of each control type, the hooks to communicate via Common Object Request Broker Architecture (CORBA) protocols, and the automation frameworks. Going along with this, an Oracle database stores all information needed to instantiate the control points, automation scripts, and the experiment configurations.

OTS LASER SYSTEM UPGRADE

As part of the NIF team's continued effort to improve our understanding of the plasma conditions, we recently deployed a diagnostic capable of measuring OTS from the imploding target. This method significantly improves the precision at which we can measure the plasma's temperature, density, and flow velocity. From this we'll gain a better understanding of how the laser interacts with the plasma, and how we can further reduce unwanted interactions. This in turn will lead to even better symmetry during the implosions and higher fusion yields. [1]

This upgrade consists of two large scale pieces: the OTS Laser (OTSL) and the OTS Diagnostic (OTSL-D). Unlike previous upgrades such as the Advanced Radiographic Capability (ARC), OTSL did not reuse any of the existing NIF beam path. Instead, we built a new room in NIF's switchyard 1 to house the front end and amplification components. The laser light follows a new beam path into the target area. Just outside the target chamber wall, we convert the laser light from the front end's 1053 nm to 211nm.

OTSL-D consists of two primary components. Frist you have the diagnostic package consisting of a spectrometer and alignment cameras. This package was specifically designed to work with either OTSL at 211 nm or a standard NIF 351 nm beamline as the probe laser. This allows us to install the diagnostic in multiple locations and inspect the target from multiple angles. In addition to the diagnostics, OTSL-D contains a separate set of alignment mirrors and cameras in a laser launch package. Due to the location where OTSL enters the target chamber, it cannot fire directly into the hohlraum's laser entrance holes. To get around this, we reflect OTSL off the laser launch mirrors and into the target.

OTS Software Upgrade Scope

To support the OTSL system, we needed to modify every layer of the ICCS system. Down at the Front-End Processor (FEP) layer, we made code changes to support multiple new device types, such as energy diagnostics,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

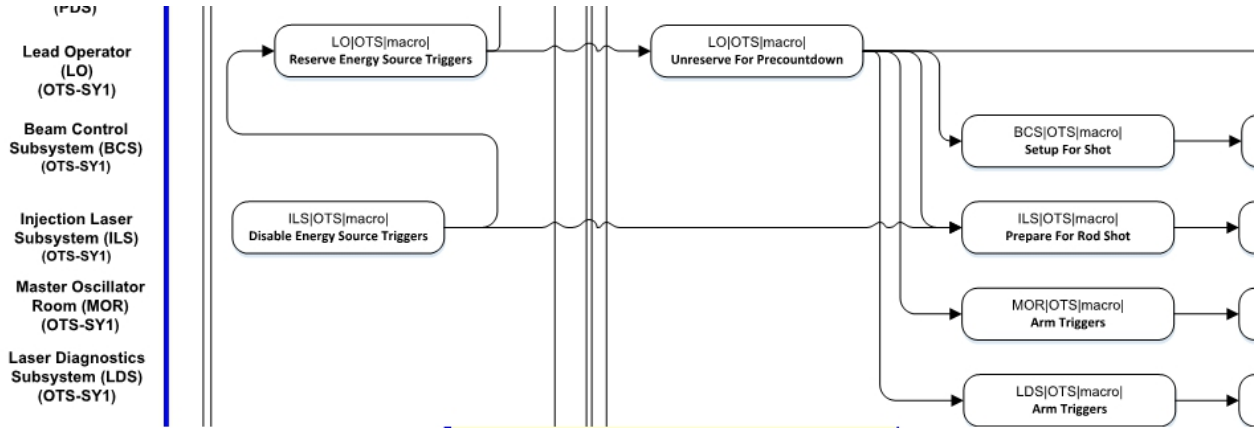


Figure 1: Snippet of OTSL shot automation swim lanes.

cameras, and power supplies. Along with the code changes, our database team needed to add new database schema and validations to support the new device types. Also in the FEP layer, we made large scale data changes to support additional instances of already supported devices.

In the next layer above, our automatic alignment team implemented a large number of new alignment loops. [2] While some of these loops were similar to existing NIF alignment loops, some operations (such as the tuning the 5ω converter crystals) required brand new approaches and image processing. Also in this layer, we needed to expand our pulse shaping system to support the new OTSL front end. Rounding this layer off, we needed to expand our status and propagation processes to handle all of the new control points.

The ICCS shot automation layer also saw significant changes. The ICCS shot team expanded their frameworks to handle this new class of diagnostic system. This resulted in 5 new automation swim lanes capable of setting up OTSL in parallel with the rest of NIF (Fig. 1). They also added multiple branching points to allow the possibility of OTSL-D being physically installed and ready for alignment operations at different points in the shot cycle. Finishing all of this off was another large data change specifying the exact details on when shot should position/verify/lock out every control point.

Last but definitely not least, multiple GUI changes were needed as part of this effort. Each of the new device types required a brand-new maintenance panel. In addition, we added new broadviews aligning with the OTSL control points diagram (Fig. 2). Several of our maintenance and commissioning tools were expanded to work with the new OTSL devices. Finally, we added tools to existing maintenance panels to assist with commissioning the system, such as real time image processing to our video displays.

MANAGING THE DATA CHANGES

Given the amount of data we deal with, the ICCS team has developed a number of tools to simplify manipulation of our databases. These tools range from very generic to highly specialized one time use tools. For the purposes of this paper, we'll be focusing on the generic tools that may provide inspiration to other teams.

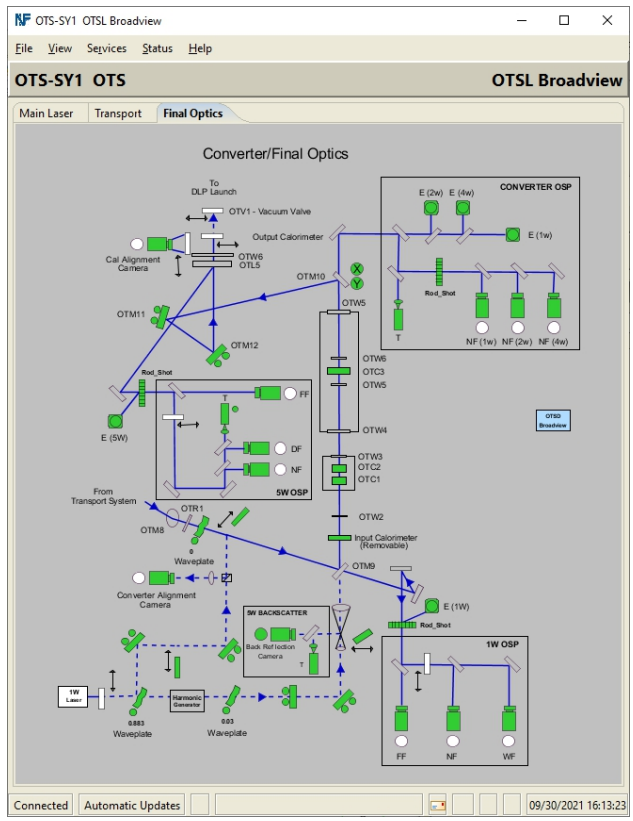


Figure 2: Broadview showing OTSL control points in the target area before entering the target chamber.

QuickMod

Originally introduced as a way for non-software developers to safely and efficiently make database changes, QuickMod rapidly became our standard method of making database changes. At the highest level, QuickMod translates Excel files into a SQL statements. Every sheet in the file can perform one of four basic operations (INSERT, UPDATE, DELETE, or CLONE)

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

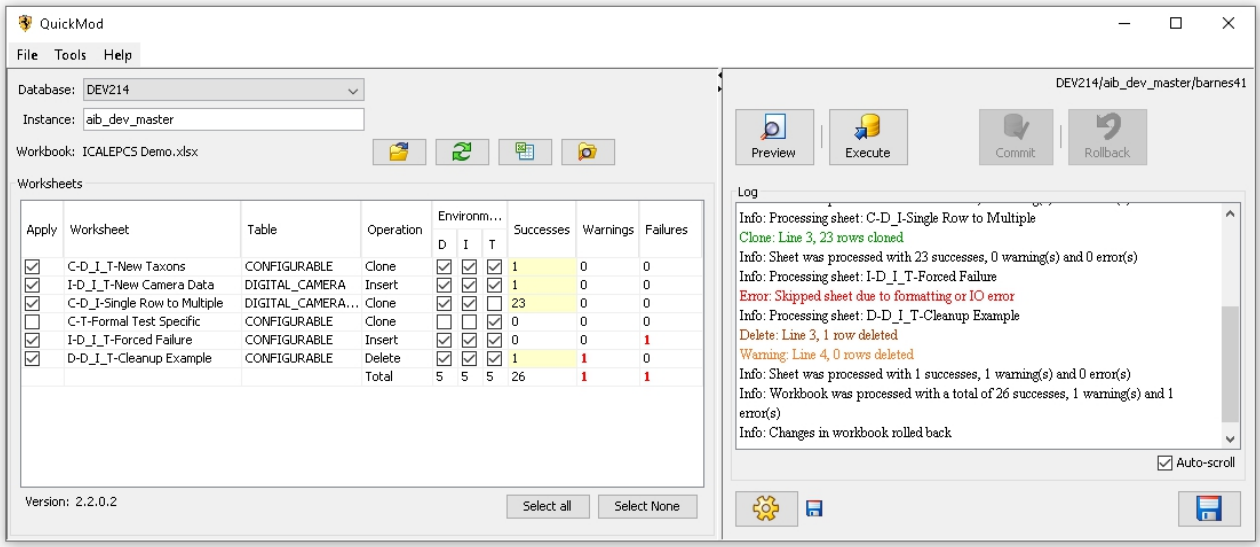


Figure 3. QuickMod GUI Interface.

on a single table. The first three operations align with the standard SQL definitions. Our clone operation combines multiple SQL statements to atomically query a table, modify the results, and insert the new rows back into the same table. QuickMod also evaluates Excel functions before applying each sheet. This allows for concise workbooks that can tailor themselves to each environment. For example, we commonly create workbooks that derive machine hostnames based upon the environment the data is being applied to (development, integration, formal test, or production). To round all this out, QuickMod provides a simple GUI interface (Fig. 3) for application of single files and a command line interface for release automation. The GUI interface provides users quick feedback on the success/failure of each operation and the ability to roll back changes before committing them to the database.

QuickMod has demonstrated several advantages. First, it provides an easy method for non-software developers to make database changes. We frequently have a software developer build an example Excel file for a subject matter expert (SME). The SME can then adjust values in the Excel file and have operations modify the database without further involvement of the software team. Second, the input files can easily be reviewed both before and after application to the database. Third, QuickMod provides a convenient place to implement validations and restrictions not easily performed by the database. For example, the ICCS architecture allows for multiple developers to work in their own personal instances while sharing the same database schema. This was done by adding an INSTANCE column to almost all database tables. Since developers rarely need to modify multiple instances at once, QuickMod restricts application of changes to a single instance at a time. Developers still have the option of falling back to SQL if they truly need to modify multiple instances simultaneously. Fourth, the granular feedback provided by the QuickMod GUI has reduced the time needed for developers to track down errors in their data changes. After executing, QuickMod provides the number of successful changes, warnings (e.g. an update that modified no rows), and errors on both a per sheet and per row basis. It also provides a log window with full details including the exact row/sheet that caused the error. Finally, by controlling the files in our configuration management system and deploying via command line automation, we ensure we deliver the entire set of intended changes to each environment.

While QuickMod is exceptionally useful tool, there are a few challenges that made it insufficient for a project the size of OTSL. The largest challenge comes from the domain knowledge needed to initially construct an input file. The ICCS configuration database contains over 1400 tables. Within each of those tables, we have a mix of columns that need to change with every entry and those that rarely change. Thus, developers frequently find themselves needing to consult domain experts both to simply identify the needed tables, as well as to identify what questions they need to ask of external teams. The next challenge comes from the quantity in data. This shows up in individual typos, copy and paste errors, as well as simply identifying when rows or sheets are missing.

Batch Cloning

To help deal with the challenges of using QuickMod alone, we created a batch cloning tool. For input, the tool takes in a list of existing control points, their new names, and new processor information. The tool first reads in all data related to the existing control points. It then adjusts the data based upon the new names and processors. Next, it removes any duplicate data. For example, when multiple digital input/outputs are defined on the same card/processor combination, it consolidates the data to a single definition of that card. As output, the tool generates an Excel file already formatted for use in QuickMod. In doing so, it highlights columns that developer likely needs to update after consulting external teams. Finally, the tool ensures that the output files will

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

create emulated devices in offline environments and real devices in production.

We ended up using the batch cloning tool for around ¾ of the OTSL data, and it quickly showed its benefits. With the first batch of OTSL control points, it cut the effort needed in half. And that time includes the effort to develop the tool. While it didn't completely eliminate the need to reach out to the domain expert for each device, it drastically reduced the amount of information needed. It also removed the possibility of missing data.

After using the tool, we also found that it introduced some new challenges of its own. Specifically, the tool did not make it easy to respond to requirements changes after the developer had started their work. The batch cloning tool excelled at getting the first 80% of the work done. After that, a developer needed to manually update the generated QuickMod file to include information gathered from other teams. Since all changes were consolidated into a single file, developers frequently would choose to manually respond to requirements changes instead of rerunning the tool and having to restart their work. The second challenge showed up when we started commissioning. Since the tool copied all data related to the source control points, it frequently copied too much data. Many of our control points have named configurations that we call setpoints. We found a number of instances where we deployed setpoints that either didn't make sense with the newly deployed control point or caused confusion since they were supposed to be created during commissioning.

Templates

Learning from our experiences with the batch cloning tool, as well as other data heavy project, the ICCS team choose to upgrade QuickMod to support templates. The templates are initially designed by a domain expert. They start with a definition sheet that contains both documentation on how to use the template and a section where each row equates to one new control point being generated when applied. After that, the Excel file contains all of the data necessary to make a single instance of a control point. Unlike a standard QuickMod input file, fields that we expect to change are replaced with template variables (such as <<name>>). When QuickMod applies the file, it will run all sheets after the definition sheet once for every new control point being added. And before each run, the template variables are replaced with the values defined in the definition sheet.

While this new system initially requires more time from a domain expert, it has shown several advantages. First, non-experts can use the templates with little to no input from the domain expert. Frequently the documentation provides enough guidance on what questions to ask other teams, and which columns the developers need to worry about changing. Second, a developer can easily respond to a change in requirements. For example, changing a control point's type frequently boils down to removing a row from one definition sheet and adding a new row in different template. Third, since the templates are essentially the data for a single instance, they become easy to generate and review. In ICCS, most of our control types have data in less than ten tables, and frequently only have one to four rows per table.

TESTING THE CHANGES

For testing changes of this magnitude, ICCS both reused existing testing frameworks (both automated and manual) and adapted some of our ancillary tools to assist in testing. This allowed us to deploy the software with high confidence that minimal adjustments would be needed in production. It also allowed us to rapidly make significant code changes when the new energy diagnostics were significantly upgraded after our initial software deployment.

Testing Device Control Changes

Our front-end processor (FEP) layer is responsible for making any piece of hardware available via a CORBA interface. We have two automated test frameworks responsible for unit and integration testing this layer. Both frameworks are built on top of JUnit and tied into our nightly build system. We designed the unit test framework to minimize the effort needed add new tests. Developers simply need to add a single line annotation to either the class or to individual test functions in order to designate a test suite. When the test framework runs, it scans the compiled code base for these annotations and dynamically builds up the test suites to run. We currently have two test suites fast and slow which respectively take approximately 2 minutes and 10 minutes to run. These tests predominantly focus on validating individual Java class.

The second framework performs black box testing via the CORBA communications layer. We introduced this framework when we ported the FEP layer to Java [3] and continue to expand it with every new control type. Given that a single CORBA interface frequently supports multiple different models of hardware (e.g. different cameras), we needed the ability to rerun the same test classes on different control points in our system. To support this, we choose to define the test suites in our configuration database. We then created a custom JUnit test runner that ties into a running instance of our system and queries the database to determine the tests that need to run. The runner also gives us the capability to run multiple test suites in parallel under different Java virtual machines. In addition to the test runner, we created a few JUnit rules to assist with common low-level operations (such as locking out control points). The framework also provides utility/base classes for testing areas where the functionality is relatively similar across control types (e.g. our interfaces for setpoints/named configurations). For a sense of scale, the test suites in this framework take between 5 minutes and 6 hours to run.

Testing Shot Automation Changes

In our offline integration environment, we predominantly rely on nightly runs of our automated shot

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

tester (AST) [3]. We currently have AST run five shots in each of our three releases under active development. To maximize our testing coverage, the experiments are specifically designed to have a mix of capabilities (such as OTSL). With each of these shots, AST will do its best to drive each shot to completion and provide a report on any failures 15 minutes before by our morning meeting. AST also provides us the capability to run shots on demand throughout the day.

In addition to AST testing, we perform a large variety of manual tests on shot changes. During integration testing of new functionality, developers will manually check all shot operations (e.g. positioning control points or verifying positions). In formal test, our QA will run multiple manual shots to confirm expected operations. They will explicitly test any verifications responsible for machine safety on a new or modified component (a different system is responsible for personnel safety checks). After deployment to production, we perform a final round of testing during the initial commissioning test shots. This testing includes manual verification that the automation layer correctly positioned all new control points, and a final verification of all machine safety checks.

COMMISSIONING THE SYSTEM

Due to the amount of new hardware with OTSL, we needed to come with a way up with a commission and tune the system without interfering with NIF's ability to continue experiments. We initially started by providing the OTSL team with an interface into the control system that could only access the OTSL control points. The team was then required to manually setup and fire OTSL. After attempting a few shots in this manner, we quickly realized the need to partially automate the process. To do this, we made use of our Target Alignment Assistant Tool (TAAT) [4]. TAAT was originally designed to allow rapid automation of alignment procedures for unique targets. As such, it reads in specially designed Excel files that define the scripted actions to take, where to ask the user for input, and where to branch if necessary. It then provides a basic user interface while it sends the corresponding CORBA commands into the system. With this, we were able to reduce the time to manually arm and archive OTSL devices during each iteration of a tuning shot from 1 hour to 5 minutes.

COMMISSIONING THROUGH COVID

With the ICCS team working almost exclusively from home, the pandemic pushed us to rethink and develop new strategies for supporting remote troubleshooting and commissioning operations. One of the strategies has worked out so well that we plan to continue using it even after the team returns on site. When a team working on NIF needs assistance, they initially setup a video teleconference and reach out to the domain experts needed. As part of the teleconference, the operations team will share the desktop of the console where assistance is needed. This allows the experts to simultaneously see what's going on in the control room, have a meeting with everyone involved, and still access tools that they have on their development machines. Building on top of this, the ICCS team maintains a Microsoft Teams channel providing summaries and status updates for any unplanned assistance requests.

The remote desktop capabilities have proven exceptionally useful in many ways. The most significant improvement is that we can have experts see the operators' screens and no longer need to rely on their descriptions of what they're seeing. This both reduces confusion and allows experts to notice small items that the operators may have overlooked or assumed were normal. This has also proven extremely useful for the alignment team, as they can now watch the alignment cameras in real time. Additionally, this functionality allows developers to remotely troubleshoot problems in labs that previously would have required them to don protective equipment and operate under escort due to safety hazards.

The strategy has also made the team more effective both when commissioning and troubleshooting operations outside of normal hours. The easy availability of teleconferencing has significantly reduced instances of multiple groups working on the same problem in parallel in their own communications silos. The Teams channel has provided management an easy way to stay appraised of problems in production. It has also improved the details available for deep dive troubleshooting the next day. Finally, the ability to work effectively from home has been a boon for SMEs that need to support shot operations well outside of their normal hours, or for extended duration shots.

CONCLUSION

The ICCS team successfully leveraged custom data manipulation tools, automated testing, and a remote work strategy to efficiently deliver controls for the OTSL system. For other teams considering similar tools and strategies, we have a few recommendations. First, focus on keeping each iteration of a tool/process change small and focused. This will lead to tools being completed faster and easier. It also keeps you from sinking significant effort on a tool that might not fully meet your needs. Second, always consider making tools more generic to allow for reuse in unforeseen circumstances. For example, when we initially wrote TAAT, we had no inkling that we would use it to automate non-alignment activities. Third, always keep an eye out for automation opportunities, especially testing.

ACKNOWLEDGEMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] P. S. Datte *et al.*, "The design of optical Thomson scattering diagnostic for the National Ignition Facility," *Rev. Sci. Instrum.*, vol. 87, p. 11E549, Nov. 2016. doi:10.1063/1.4962043

[2] A. Awwal *et al.*, "Image Processing Alignment Algorithms for the Optical Thomson Scattering Laser at the National Ignition Facility," presented at ICALEPCS'21, Shanghai, China, paper WEAL01, this conference.

[3] B. T. Fishler *et al.*, "Sustaining the National Ignition Facility (NIF) Integrated Computer Control System (ICCS) over its Thirty Year Lifespan," *in Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1201-1205. doi:10.18429/JACoW-ICALEPCS2017-THCPL06

[4] M. Fedorov *et al.,* "New Visual Alignment Sequencer Tool Improves Efficiency of Shot Operations at the National Ignition Facility (NIF)," *in Proc. ICALEPCS'17,* Barcelona, Spain, Oct 2017, pp. 328-333. doi:10.18429/JACoW-ICALEPCS2017-TUMPA01

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV024

VSCODE-EPICS, A VSCODE EXTENSION TO ENLIGHTEN YOUR EPICS CODE

Victor Nadot†, Alexis Gaget, Francis Gohier, Françoise Gougnaud, Paul Lotrus, Stéphane Tzvetkov, *IRFU, CEA, Université Paris-Saclay, F-91191 Gif-sur-Yvette, France

Abstract

vscode-epics is a Visual Studio Code extension developed by CEA Irfu that aims to enlighten your EPICS code. The development was originally initiated by one of our students in 2017.

This module makes developer life easier, improves code quality and helps to standardize EPICS code.

It provides syntax highlighting, snippets and header template for EPICS files and snippets for WeTest[1].

vscode-epics module is based on Visual Studio Code language Extension and it uses basic JSON files that make feature addition easy.

The number of downloads increases, version after version, and the various feedbacks we receive motivate us to strongly maintain it for the EPICS community. Since 2019, several other laboratories of the EPICS community have participated in the improvement of the module and it seems to have a nice future (linter, snippet improvements, specific language support, …).

The module is available for free on Visual Studio Code marketplace[2] and on EPICS extension GitHub[3]. CEA Irfu is open to bug notifications, enhancement suggestions and merge requests, in order to continuously improve vscode-epics.

FEATURES

The syntax highlighting (Figure 1) provides the function-ality to detect the EPICS keywords for a large scope of EP-ICS files (“.db”, “.template”, ".substitutions”, “.cmd”, “.st”, “.proto”, “.c”). As a concrete example, record types, record fields, macros and comments are supported for “.db” files.

Figure 1: With and without syntax highlighting.

Snippets are yet an even more powerful feature of the extension (see Figure 2). It makes it possible to generate any records with the basic EPICS fields. They are very val-uable for new EPICS developers.

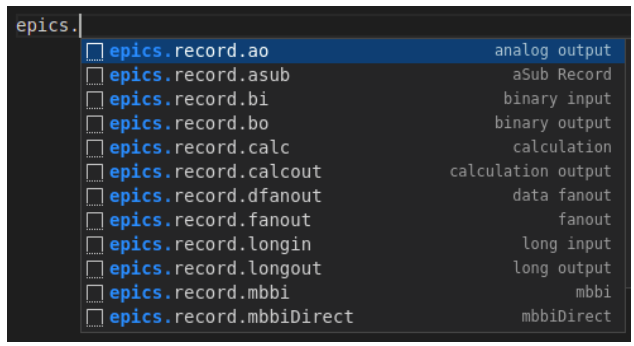


Figure 2: Scrollable list of available snippets.

Another interesting feature is the header template gener-ation (see Figure 3). Largely inspired from our collabora-tion with ESS, the headers are composed of the following fields:

1. “file”: the file name is automatically written at the top of the header. This is useful when you use substitution files that generate a large “.db” file with all your templates inside. Thus, you can easily see the different template demarcations in the generated “.db” file.
2. “author”: the author and company references.
3. “description”: a short presentation describing the file scope.
4. “macros” :in order to improve code readability, it is strongly advised to add the list of the macros used in the file with a short description. They are split in two types. Mandatory: they are required to be defined at higher level to use the template. Optional: they have default values so they are not required to be defined at higher level to use the template;

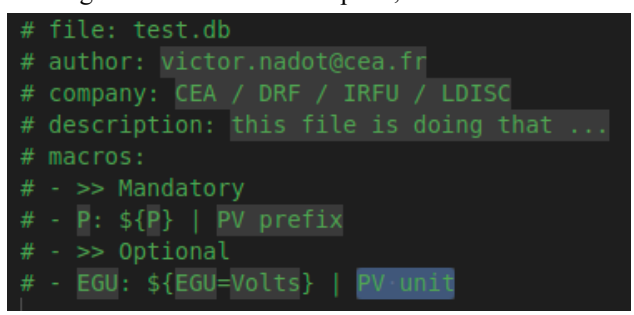


Figure 3: File header provided by vscode-epics extension.

More snippet can be added as long as the language you are using is part of the EPICS extensions[4]. For instance WeTest, a software developed by CEA/Irfu to automate acceptance tests and shared with EPICS extension, has snippets integrated to epics-vscode module.

Besides making developer life easier, vscode-epics extension improves your code quality and code homogeneity

* † victor.nadot@cea.fr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV024

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

across your developments. Indeed, the snippets feature will suggest the most used EPICS fields by record (they are not absolutely required but they help to improve the overall quality of your EPICS database). Here is a concrete example: an "ao" snippet would suggest to fill limits (DRVL/DRVH), unit (EGU), and precision (PREC). Note that for all snippets, the description (DESC) field is also suggested. All of these fields are not mandatory to make your EPICS database work, but it really improves it, especially from the client side: units can be displayed on GUI, GUI widgets can use PV limits, etc.

vscode-epics extension is based on JSON files in order to describe its different features. It makes it easy to read, to understand the implemented features and to add new ones.

EXTENSIONLIFE

The number of downloads increases quite linearly (see Figure 4), version after version, and the various feedbacks we receive motivate us to strongly maintain it for the EPICS community.

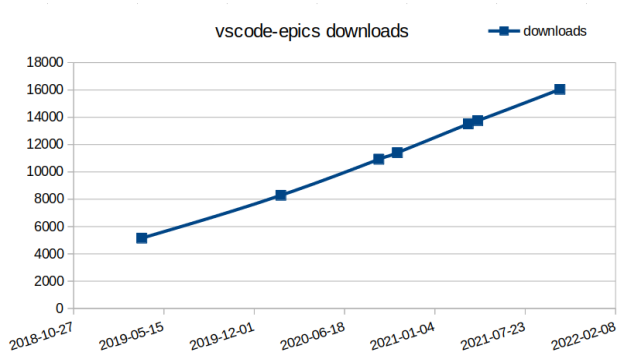


Figure 4: Vscode-epics extension downloads over the years.

Since 2019, some laboratories of the EPICS community have participated in the improvement of the module with significant contributions made by (see Figure 5):

- Institute of Plasma Physics (IPP CAS)
- European Spallation Source ERIC (ESS)
- Stanford Linear Accelerator Center (SLAC)
- Diamond Light Source (DLS)

It really is encouraging to receive some contributions from the EPICS community.

Figure 5: Contributions over the years.

The module is available on Visual Studio Code marketplace and on EPICS extension GitHub. CEA Irfu is open to bug notifications, enhancement suggestions and merge requests to continuously improve vscode-epics.

The current release is 1.1.0. The changelog (and the git history) is tracing the whole code history. In order to manage issues, I use the GitHub "Projects". Its kanban provides a good view of the tickets (see Figure 6).

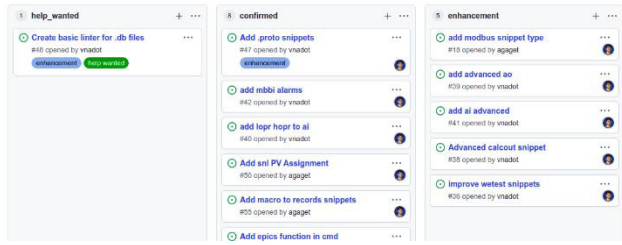


Figure 6: Vscode-epics kanban on Github.

FUTURE DEVELOPMENTS

There are few features in the pipelines, prioritized in two categories: confirmed and enhancements. There is also a "help wanted" column that contains the linter enhancement. The vscode-epics linter requires a bigger development so any help from the EPICS community, especially from someone with a linter development experience, would be appreciated. In addition, some automatic tests will be needed to avoid regressions.

Recently a colleague from Irfu CEA started a similar project for Vim (and NeoVim): epics-syntax.vim[5]. vscode-epics extension JSON files were a useful starting base. However, for now, only syntax highlighting is supported. In the longterm the same features as vscode-epics should be supported.

CONCLUSION

vscode-epics extension improves the EPICS code homogeneity and makes EPICS developers' life easier thanks to its syntax highlighting, snippets and other features.

The extension has reached a stable state for several months now. The code is available on EPICS extension GitHub repository, it is open to merge requests and issue opening. GitHub kanban makes it easy to manage, and it provides the EPICS community with a good overview of the current project status.

REFERENCES

[1] WeTest, https://github.com/epics-extensions/WeTest

[2] vscode-epics extension, https://marketplace.visualstu-dio.com/items?itemName=nsd.vscode-epics

[3] vscode-epics code, https://github.com/epics-exten-sions/vscode-epics

[4] EPICS extensions on github, https://github.com/epics-exten-sions

[5] epics-syntax.vim, https://github.com/minijackson/epics-syn-tax.vim

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV025

TangoGraphQL: A GraphQL BINDING FOR TANGO CONTROL SYSTEM WEB-BASED APPLICATIONS

J.L. Pons, European Synchrotron (E.S.R.F), Grenoble, France

Abstract

Web-based applications have seen a huge increase in popularity in recent years, replacing standalone applications. GraphQL provides a complete and understandable description of the data exchange between client browsers and back-end servers. GraphQL is a powerful query language allowing API to evolve easily and to query only what is needed. GraphQL also offers a WebSocket based protocol which perfectly fits to the Tango event system. Lots of popular tools around GraphQL offer very convenient way to browse and query data. TangoGraphQL is a pure C++ http(s) server which exports a GraphQL binding for the Tango C++ API. TangoGraphQL also exports a GraphiQL web application which allows to have a nice interactive description of the API and to test queries. TangoGraphQL has been designed with the aim to maximize performances of JSON data serialization, a binary transfer mode is also foreseen.

INTRODUCTION

Today, at the ESRF [1], we use mainly Java standalone applications for the accelerator control system. These applications are built on top of the Java Swing ATK framework [2] and Tango java APIs. Today, regarding GUI technologies, we see almost only development around web technologies such as React, Angular, Vue.js, Bootstrap, Material UI, etc... It is natural that we migrate our GUIs to web based applications. Java ATK is based on the Model View Controller model. React (Facebook) offers a very convenient way to implement this model using hooks. GraphQL [3], initially developed by Facebook in 2012, was moved as open source to the GraphQL foundation. A JavaScript Tango Web ATK built on top of React and GraphQL is currently under development at the ESRF. This framework is designed in order to ease as much as possible the migration of our Java applications.

ARCHITECTURE

MVC Model using React

React function components offer hooks that can be used to implement the MVC model. The key idea is to use the state hook and to pass the setState function handle as a dispatcher to the model. A layoutEffect hook handles the subscription in the listener list of the model as shown in Fig. 1.

The model (the GraphQL client) makes access to Tango devices using the TangoGraphQL server either through basic HTTP requests or through WebSocket as shown in Fig. 2.

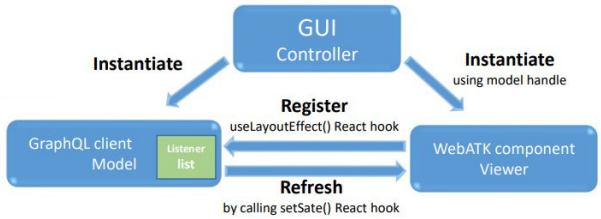


Figure 1: MVC using React.

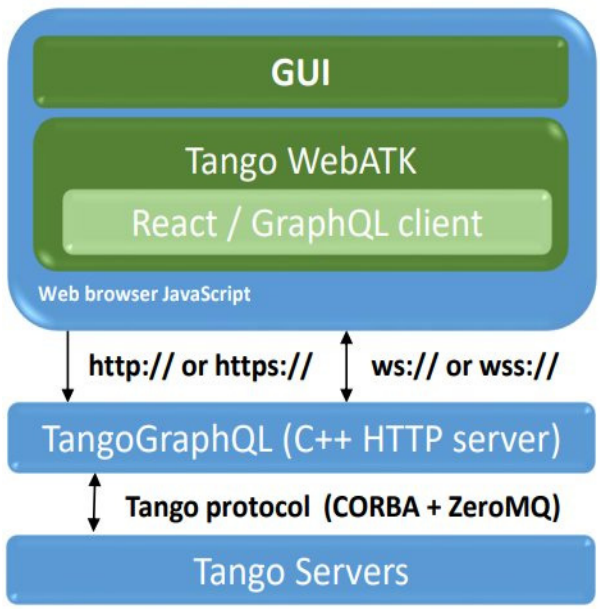


Figure 2: Architecture.

GRAPHQL

GraphQL vs REST

Compared to a REST API, GraphQL is a true query language allowing clients to query only what is needed in a single HTTP request. GraphQL also provides an introspection system giving information about the supported schema. GraphQL uses GraphQL query to perform introspection. The GraphQL foundation provides a web application called GraphiQL, based on this introspection system, enabling users to write query using modern tools such as completion, syntax checking and documentation browsing. It uses the Tango GraphQL schema definition [4] to provide all these features to the Tango GraphQL API. TangoGraphQL C++ server also exports a GraphiQL interface.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 3: WebATK Panel.

GraphQL Tango API

The GraphQL Tango API must provide all calls needed to build any applications including generic applications such as ATKPanel. WEBAtkPanel shown in Fig. 3 makes an introspection of the Tango device to display all its attributes and commands.

GraphQL defines 3 types of request: query, mutation and subscription. Queries are read only requests (basically a Tango read_attributes call), mutations are write requests (write_attributes or command_inout) and subscriptions are used to register to Tango events. The requests are sent to the server via a HTTP POST request.

When registering to an event, the TangoGraphQL server will push JSON frames when a Tango event is sent by the Tango server. When using the GraphQL over WebSocket protocol [5], the client (and the server) use the socket in a bidirectional mode (full duplex), which means that the client (or the web server) can send or receive requests at any time.

PERFORMANCE

Error Management

The GraphQL error management standard is not very convenient. When a node cannot be returned, the GraphQL standard impose that the node must be null and an additional errors JSON block has to be filled with errors. With large queries which may contain lots of errors, the browsing of this additional error tree is a bit heavy. TangoGraphQL server handles Tango errors as queries for best performance and keep GraphQL standard for parsing errors.

Binary Transfer

A GraphQL internal parser has been written from scratch and designed to support a true binary JSON transfer currently under development. Binary transfer is not a part of the GraphQL and JSON standard but it can be added without breaking the GraphQL compatibility by using HTTP header. Despite the fact that TangoGraphQL uses fast floating point serialization algorithm Grisu2 [6] from Florian Loitsch, image and spectrum need to be transferred in binary format in order to reach good performance at both server and client side. On the client side, binary transfer can be easily achieved using native JavaScript DataViews and ArrayBuffer. Table 1 shows timing measurement of the TangoGraphQL C++ server for a spectrum of 16383 random double values, the JSON encoding in text format is done using Grisu2.

Table 1: TangoGraphQL Server Performance

	TEXT, 16digits	**BIN, 64bits**
Get HTTP request	1ms	0.9ms
Tango reading	0.9ms	0.9ms
JSON encoding	6ms (Grisu2)	0.3ms
Send HTTP response	0.4ms	0.3ms

Asynchronous Group Calls

When the client needs to retrieve data from several devices (typically the state of n devices), the client can construct a GraphQL request using labeling. The server can detect labeled read_attributes calls and launches parallel asynchronous calls using Tango Group calls.

Load Balancing

TangoGraphQL is also a Tango server and can be configured, monitored or instantiated using standard Tango tools. It also has attributes that indicates number of connected clients, number and type of connections, network transfer, etc… The client can use this information to select the less loaded instance among a set of running TangoGraphQL servers as shown in Fig. 4.

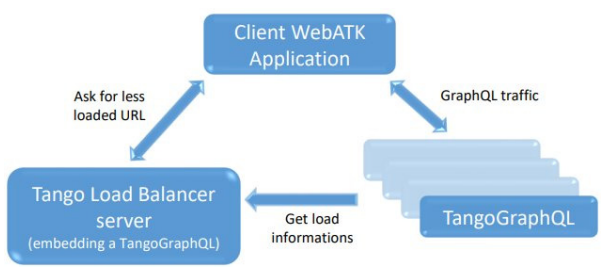


Figure 4: Load balancing model.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV025

TCP Connection

TangoGraphQL server uses multi-threaded connection handling and is HTTP 1.1 compliant. It uses the fact that in HTTP 1.1 the TCP connection can be persistent (keep-alive). The server keeps a "device factory" during all the TCP connection life cycle. This prevents from re-importing and re-introspecting each time Tango devices and avoids unwanted overhead with the Tango database and devices. When the connection is closed, TangoGraphQL releases all allocated resource associated to the connection.

CONFIGURATION

TangoGraphQL server can be configured using Tango properties:

- *Port*: HTTP server port (default is 8000)
- *IdleTimeout*: Timeout before closing a HTTP connection (default is 60s)
- *Authentication*: None or Basic
- *Users*: List of allowed users, (List of userName: sha1Password)
- *AllowMutation*: Allow mutation mode (ALWAYS, AUTHENTICATED, NEVER).
- *exportGraphiQL*: True to enable GraphiQL (available via http(s)://host:port/graphiql/)
- *httpsEnable*: Enable HTTPS server (need certificate)
- *httpsCertificate*: Link to the certificate file cert and its privKey file. (The server needs read access to this 2 files)
- *ResolveClientIP*: Resolve names in (connected) clients attribute

AUTHENTICATION AND ACCESS CONTROL

TangoGraphQL uses Open SSL for https with basic HTTP authentication scheme. Single SignOn authentication scheme is currently under development. Tango Access Control cannot be implemented using Tango 9 as it is not possible to set a username for a Tango client thread. This will require new feature of Tango 10.

CONCLUSION AND DISCUSSIONS

After an investigation done at the ESRF of the 4 present solutions (including the one presented in this paper) for a Tango backend, GraphQL appears to be the best alternative.

- Tango Rest API (java REST API on top of Apache) [7]
- RestDS (& RestDS2) (C++ REST API on top of boost HTTP server) [8]
- MaxIV TangoQL (Python GraphQL on top of AIOHTTP & Graphene) [9]

In this paper, a C++ implementation of a Tango GraphQL schema is presented and express the minimum requirements we have at the ESRF in order to make the transition from our Java applications to Web applications possible. The GraphQL schema is not fixed; it may evolve over time, using deprecations, for example. The schema proposed here is still open to discussion. The goal is for the community to converge on a common GraphQL specification which can be supported by multiple implementations.

REFERENCES

[1] European Synchrotron Radiation Facility, https://www.esrf.fr

[2] Java ATK framework, https://gitlab.com/tango-controls/atk

[3] GraphQL Foundation, https://graphql.org/

[4] Tango C++ GraphQL GitLab, https://gitlab.com/tango-controls/TangoGraphQL

[5] GraphQL over WebSocket Protocol, https://github.com/enisdenjo/graphql-ws

[6] Florian Loitsch, "Printing floating-point numbers quickly and accurately with integers", in Proc. 31st ACM SIGPLAN Conference on Programming Language Design and Implementation, June 2010, pp. 233-243.

[7] Java TANGO REST API, https://tango-rest-api.readthedocs.io/en/latest

[8] Sedykh, G.S *et al.*, "The C++ TANGO REST API Implementation". Phys. Part. Nuclei Lett. 17, 604–606 (2020). https://doi.org/10.1134/S1547477120040391

[9] TangoGQL, MaxIV, https://developer.skao.int/projects/web-maxiv-tangogql/en/latest/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV026

INTEGRATING OPC UA DEVICES IN EPICS

R. Lange†, ITER Organization, 13067 St. Paul lez Durance, France
R. A. Elliot, K. Vestin, European Spallation Source, 221 00 Lund, Sweden
B. Kuner, C. Winkler, Helmholtz-Zentrum Berlin, 14109 Berlin, Germany
D. Zimoch, Paul-Scherrer-Institut, 5232 Villigen PSI, Switzerland

Abstract

OPC Unified Architecture (OPC UA) is an open platform independent communication architecture for industrial automation developed by the OPC Foundation. Its key characteristics include a rich service-oriented architecture, enhanced security functionality and an integral information model, allowing to map complex data into an OPC UA namespace.

With its increasing popularity in the industrial world, OPC UA is an excellent strategic choice for integrating a wealth of different COTS devices and controllers into an existing control system infrastructure. The security functions extend its application to larger networks and across firewalls, while the support of user-defined data structures and fully symbolic addressing ensure flexibility, separation of concerns and robustness in the user interfaces.

In an international collaboration, a generic OPC UA support for the EPICS control system toolkit has been developed. It is used in operation at several facilities, integrating a variety of commercial controllers and systems. We describe design and implementation approach, discuss use cases and software quality aspects, report performance and present a roadmap of the next development steps.

INTRODUCTION

Open Platform Communications Unified Architecture (OPC UA) is an open standard architecture intended to improve and expand interoperability in the Industrial Automation industry. It is a machine-to-machine communication protocol for industrial automation developed by the OPC Foundation [1] and released in 2008.

While its predecessor, OLE for Process Control (OPC; re-branded as OPC Classic) was developed by Microsoft and used a transport layer based on proprietary Microsoft software (e.g., OLE, DCOM), OPC UA focuses on platform independence and uses well-known open standards like TCP and TLS.

Over the last years, OPC UA has become increasingly popular in the world of Industrial Automation. Across all vendors, framework and device types, it is a key word for interoperability and integration, boosted by the OPC Foundation's certification program that ensures high levels of compliance.

OPC UA ARCHITECTURE

Features

OPC UA integrates all functionality of the OPC Classic specifications into a single extensible framework, providing (see [1]):

- Functional equivalence: all OPC Classic specifications are mapped to Unified Architecture.
- Platform independence: hardware and operating system portability covers everything from embedded systems to cloud-based infrastructure.
- Security: encryption, signing, authentication and auditing allow messages to be transmitted securely with verifiable integrity.
- Extensibility: new features can be added without affecting existing applications.
- Information modelling: a framework helps defining complex information.

Core Concepts

The client-server communication model of OPC UA is layered: on top of a *transport*, which can be secured, the client opens a *session* to a server. Within that session, every piece of information is an *item* that can be written or read. Items are uniquely identified by a *nodeID*, consisting of a *namespace* number and an *identifier*, which can be numerical or a name string. To overcome the disadvantage of having to look up nodes by their name with every access, the client can *register* nodes with the server, to allow optimizing their name resolution.

Each item has a typed value, which can be of a basic data type, an array thereof, a union, or a structure consisting of multiple named elements, which are themselves of basic type, arrays, unions or structures.

Opening *subscriptions* in the session allows monitoring items for changes in their values. The subscription has a configurable *publishing period*, but any monitored item in the subscription can use a different *sampling period* for updating the value from the underlying device. For values that are sampled faster than their subscription is published, data loss can be avoided by defining a server-side queue.

OPC UA *methods* are basically remote procedure calls. They allow the client to send parameters (which can be of structured data types) and can return results to the client. All handshake and synchronization is part of the protocol specification and done within the client and server libraries.

† ralph.lange@iter.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV026

INTEGRATION OF CONTROLLERS

In the existing approaches to integrate industrial controllers in control systems, a number of issues can be identified that can benefit from the application of OPC UA.

Industry Standard

Integration of industrial controllers in control systems is a field of mostly proprietary protocols, which are usually vendor and sometimes device series specific. Setting up the communication with a new controller from a different vendor means switching to (or developing) a new driver, dealing with the issues and subtleties of a new protocol, gathering knowledge with a very narrow scope. For facilities with a long life time and in-kind contributions, the accumulation of driver software eventually induces a heavy maintenance burden.

Using an industry standard greatly reduces the amount of software related to the integration. Knowledge of the protocol is wide spread. Independent certification lowers the risk of particular devices behaving unexpectedly.

Symbolic Addressing

In some of the controller communication protocols, low level addressing is done through byte offsets in data blocks inside the controller, e.g. for the Siemens S7 communication [2, 3] and the TCP data block send/receive [4]. Such offset addressing is brittle and susceptible to misalignments between the client and the controller. Several configuration approaches include the generation of both the data blocks on the controller and the configuration of the client as well as strict version tagging and checking to reduce the risk of mismatch [5, 6].

Symbolic addressing makes the client configuration much more obvious and handles modification of the controller software in a robust fashion. Controller and client development workflows can be cleanly separated and have a clear interface.

Structured Data

Modern industrial controller programming uses object-like blocks that are repeated when systems are scaled up. Such programming techniques inherently produce hierarchical data structures on the controller. Mapping these objects into a flat structure for the controller communication (e.g., for the send/receive driver [4]) needs additional programming on the controller that adds complexity, needs development resources and uses run time resources on the controller.

Using OPC UA, the controller structures are directly mapped into the OPC UA namespace. No additional programming is needed on the controller. The client configuration directly uses the server-side hierarchy.

Remote Procedure Calls

Existing communication protocols are usually data-centric – they cover reading and writing data from/to the controller memory. Many control systems integrations have a concept of commands, which needs to be implemented on top of the data-centric interface. Such implementations require additional effort and programming on both the client and the controller for the synchronization (handshake).

Industrial controllers map the OPC UA *methods* feature (see below) to the execution of a function block, including the transmission of parameters and return values. This allows for a very simple yet rich and powerful implementation of commands.

Security

Specific applications, e.g. in the context of medical accelerators and for communications with safety and interlock systems, require or benefit from using the security features of OPC UA.

Secure OPC UA communication also allows running connections over insecure or open networks, e.g. wireless or remote connections.

Embedded OPC UA Servers

Some popular standard PLCs, like the Siemens S7 1200 and 1500 series, can run OPC UA servers embedded inside the controller. This allows for a slim and simple integration architecture, avoiding additional layers of hard- and software between controller and integration.

EPICS DEVICE SUPPORT

From the beginning of the project, the goal was to provide a solution completely based on free and open software.

However, based on an evaluation of existing OPC UA clients, the C++ Based OPC UA Client SDK by Unified Automation [7] was selected as the first client library to be supported. Other candidates were disregarded because of incomplete implementation of the OPC UA specification, low robustness in failure scenarios or poor consistency of their API with EPICS concepts.

Consequently, the OPC UA Device Support module is also written in C++, using an architecture that supports adding other low level client libraries. A second implementation is currently being added to the project, based on the client library of the open62541 [8] project.

Layered Structure

The interface to the EPICS Database supports all applicable record and data types from EPICS Base. In addition to the configuration of sessions and subscriptions, a set of tracing and debugging functions is available to be used from the EPICS iocShell.

Below that, a generic layer of interfaces (C++ virtual base classes) follows the core definitions of OPC UA: session, subscription, item and data element (implementing one value inside a structure) are interfaces to their respective implementations by the client library. A set of helper classes (implemented as templates) covers functionalities that all client libraries need to provide.

The lower layer provides implementation and adaptation of the generic APIs to the specifically used client library. In the case of the Unified Automation client, this part con-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV026

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

tains a pretty straightforward mapping to the SDK's interfaces. The open62541 based implementation adds a fair amount of complexity in this layer, wrapping around the simpler C APIs of the open62541 client.

Recently, a second implementation of an EPICS Device Support for the open62541 client has been announced [9], suggesting that this client in its current state is a viable alternative.

Robust Support for Structured Data

Structured data needs to be well supported, as the Siemens S7 embedded OPC UA server (one of the main use cases) performs much better when using structured data compared to addressing every element as a separate OPC UA item.

Interfacing structured data to the EPICS Database pushes for a separation between the item with its configuration parameters and the elements (values) of the structure that are each connected to a different EPICS record. This is reflected in the implementation: A single item record (an additional EPICS record type) carrying the item related configuration connects to the item object below. Many regular EPICS records carrying the single value related information and their address relative to the structure root connect each to one data element object below. The addressing information from the data element records is used to create a tree of data element objects that represents the expected structure. As the OPC UA server may change its internal structure between sessions, this IOC side tree must be mapped against the structure that is actually found on the server every time a session is established or re-established.

Writeable data structures can be handled in two modes: When many elements of a structure are changed in one moment (e.g., when loading back snapshot data), OPC UA sending can be triggered explicitly by writing to a field of the item record. When only single elements of a structure are written at a time (e.g., while operating), OPC UA sending can be triggered automatically whenever a data element is written to.

SOFTWARE QA

Static Code Analysis

Static code analysis is being performed on the module as part of the Continuous Integration setups, making sure that the code is consistently showing high quality ratings. The public service Codacy is used for static analysis of the code in the public upstream repository [10] in addition to ITER's local installation of SonarQube [11].

Unit Testing

Unit tests have been introduced for the helper classes in the generic layer. Coverage, however, is still very limited. Unit tests for the lower level code would be highly desirable, but will need a noticeable investment. A possible approach would be using a mock of the client library APIs with GoogleTest [12] to allow separate testing of the integration layer.

End-to-End Testing

An end-to-end test suite was developed to provide confidence that the device support module functions as expected, in terms of correctness and performance. It also serves to test for any regression that may be introduced, as the tests can be triggered to automatically run for all new commits to the source repository.

The test setup comprises of a software OPC UA server, implemented using open62541 [8], two IOCs, and a Python-based test suite using pytest [13]. In order to communicate with both an EPICS IOC and the OPC UA software server directly, both the PyEpics [14] and opcua [15] Python modules are used. Running of the software server and IOC is handled during test execution by way of test setup and teardown methods.

The OPC UA test server hosts a set of items, covering all supported OPC UA data types. *Connection tests* validate the ability of the EPICS module to connect, reconnect and disconnect from the OPC UA server under a variety of conditions. *Variable tests* verify that that EPICS module correctly translates variable values to/from the types used on the OPC UA server. *Performance tests* are used to characterize the performance of the EPICS module in terms of execution time and memory consumption. *Negative tests* ensure that the EPICS module is able to handle error cases and incorrect inputs gracefully.

The test suite is currently in active use at the ESS facility to provide regression testing under the localized EPICS build system, ensuring that any changes to EPICS environment or OPC UA module have no detrimental impact on the relevant devices in the control system.

A second set of automated tests based on the same framework has been defined for the ABB Power SCADA integration at the ESS to allow efficient regression testing on the final application level as well as on the device support level.

Both test suites are configured to be executed as part of the ESS build pipeline for any new merges against the device support or application modules.

A COLLABORATIVE EFFORT

In 2016, B. Kuner at the Helmholtz-Zentrum Berlin evaluated available public and commercial OPC UA client libraries and settled on the C++ Based OPC UA Client SDK by Unified Automation [7]. He created a prototype of the EPICS Device Support [16], robust enough to be used in production at BESSY II and ITER for several years.

After performance measurements using the prototype at ITER were showing solid results, R. Lange started the full implementation [17] in 2018, based on the design outlined above. Other institutes were doing evaluations of the module while it was being developed.

In 2020, R. A. Elliot and K. Vestin began working on the end-to-end test suite described above, which will be merged to become part of the main project in the next version.

In 2021, C. Winkler was helping to finalize and test the Windows build of the Device Support. D. Zimoch and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV026

C. Winkler started the integration of the open62541 client library, which is still ongoing and is likely be added to the main project soon.

CURRENT STATUS AND ROADMAP

The EPICS OPC UA Device Support module is stable, mature and robust. It is used by several institutes against different OPC UA servers in productional setups. Table 1 shows a list of currently known applications of the module and their status.

Table 1: Existing Users and Applications (incomplete)

Facility	OPC UA Server	Status
ASIPP	LabVIEW	production
	PLC Siemens S7-1500	production
Australian Synchrotron	PLC Siemens S7-1500F	near production
BESSY II @HZB	PLC Siemens S7-1500	production
	Phoenix Contact	production
	Softing uaGate	production
CHIMERA @CCFE	PLC Siemens S7-1500	development
	LabVIEW	development
ESS	PLC Siemens S7-1500F	production
	ABB Power SCADA	near production
	Siemens DESIGO	development
Fermilab	Kepware KEPServerEX	testing
	PLC Siemens S7-400	development
IPR	PLC Siemens S7-1500	testing
ITER	PLC Siemens S7-1500	production
	Siemens WinCC OA	production
	PCVue	production
KATRIN @KIT [18]	LabVIEW	prototyping
PSI	PLC Siemens S7-1500	development
Varian ProBeam[1]	PLC Siemens S7	production
	PLC Beckhoff	production

A requirement specification exists and is kept up-to-date [19]. The remaining feature from this document that needs to be implemented is the support for OPC UA methods – this effort has been started.

Comprehensive user level documentation still needs to be added. The existing "cheat sheet" documentation is sparse.

The unit tests and the end-to-end test suite need to be extended and completed to achieve a better level of test coverage.

[1] On Windows and VxWorks platforms.

REFERENCES

[1] OPC Foundation, https: //opcfoundation.org

[2] What properties, advantages and special features does the S7 protocol offer?, https://support.industry.siemens.com/cs/document/26483647/what-properties-advantages-and-special-features-does-the-s7-protocol-offer-

[3] S. Marsching, "A New EPICS Device Support for S7 PLCs", in *Proc. 14th Int. Conf on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2013)*, San Francisco, CA, USA, Oct. 2013, paper THPPC027, pp. 1147-1149.

[4] EPICS S7PLC Driver, http://epics.web.psi.ch/software/s7plc

[5] S. Pande *et al.*, "CODAC Standardisation of PLC Communication", in *Proc. 14th Int. Conf on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2013)*, San Francisco, CA, USA, 2013, paper THPPC004, pp. 1097-1099.

[6] G. Ulm *et al.*, "PLC Factory: Automating Routine Tasks in Large-Scale PLC Software Development", in *Proc. 16th Int. Conf on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2017)*, Barcelona, Spain, Oct. 2017, paper TUPHA046, pp. 495-500. doi:10.18429/JACoW-ICALEPCS2017-TUPHA046

[7] Unified Automation, C++ Based OPC UA Client SDK, https://www.unified-automation.com/products/client-sdk/c-ua-client-sdk.html

[8] open62541 project, https://open62541.org

[9] S. Marsching, "Integrating OPC UA Devices into EPICS Using the Open62541 Library", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, China, Oct. 2021, paper TUBR05, this conference.

[10] Codacy page of the OPC UA module, https://app.codacy.com/gh/epics-modules/opcua/dashboard

[11] SonarQube, https://www.sonarqube.org

[12] GoogleTest/GoogleMock, https://google.github.io/googletest

[13] pytest, https://docs.pytest.org

[14] EPICS Channel Access for Python, https://pypi.org/project/pyepics

[15] python-opcua project, https://github.com/FreeOpcUa/python-opcua

[16] opcUaUnifiedAutomation project (prototype), https://github.com/bkuner/opcUaUnifiedAutomation

[17] EPICS OPC UA Device Support project, https://github.com/epics-modules/opcua

[18] J. Mostafa *et al.*, "Interfacing EPICS and LabVIEW Using OPC UA for Slow Control Systems", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, China, Oct. 2021, paper TUPV011, this conference.

[19] EPICS OPC UA Device Support Software Requirements Specification, revision 1.1, https://docs.google.com/document/d/1_NaSPZNGuRRt8m92Nd2NvJ6LrNAN9UkvTSSIpogSGL4

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV027

THE EVOLUTION OF THE DOOCS C++ CODE BASE

L. Fröhlich*, A. Aghababyan, S. Grunewald, O. Hensler, U. F. Jastrow, R. Kammering, H. Keller†, V. Kocharyan, M. Mommertz, F. Peters, A. Petrosyan, G. Petrosyan, L. Petrosyan, V. Petrosyan, K. Rehlich, V. Rybnikov, G. Schlesselmann, J. Wilgen, T. Wilksen,
Deutsches Elektronen-Synchrotron DESY, Germany

Abstract

This contribution traces the development of DESY's control system DOOCS from its origins in 1992 to its current state as the backbone of the European XFEL and FLASH accelerators and of the future Petra IV light source. Some details of the continual modernization and refactoring efforts on the 1.5 million line C++ code base are highlighted.

INTRODUCTION

DOOCS [1, 2] started out at DESY in 1992 as a control solution for vacuum devices – ion getter pumps and similar equipment – for superconducting cavity test stands. Later it was ported to the HERA [3] proton storage ring to replace an older predecessor. It was built around Sun Microsystem's remote procedure call protocol (*SunRPC*) that continues to be used today in many well-known services such as the Network File System (NFS) under the newer name Open Network Computing RPC (*ONC RPC* [4]).

At the time, object-oriented programming was quickly establishing itself as the dominant programming paradigm, which lead to the new system being called the "distributed object-oriented control system", DOOCS. It was implemented in C++, the natural choice for developing reliable software that could use high-level abstractions, work with limited resources, and access hardware efficiently. Java, the next "big" language to champion object-orientation, would not be released to the public until 1994/1995 [5].

The C++ of the year 1992 was quite different from modern versions of the language. It was six years before the first ISO standardization of the language [6], and many features taken for granted today were still in their infancy: Templates, namespaces, and exception handling had just been specified [7] but were not or only partially available on the compilers of the time. The standard template library (STL) and even the `bool` type would not be standardized until 1998. It is therefore only natural that early DOOCS sources, although heavily using classes and inheritance, look more like low-level C than modern C++ from today's perspective.

These first years established the backbone of DOOCS and formed many of the conventions that shaped the development of the code over the following two decades. Although the code was continually extended and maintained, its basic style changed relatively little until some years after the introduction of C++11 [8]. Although the new standard could not be adopted for the core libraries until 2018 due to lack of compiler support on the Solaris platform, it lead to renewed interest in C++ and modern programming styles and attracted new developers. Gradually, more and more effort was put into the modernization of the code base. We are going to highlight a few of the changes made during this ongoing modernization effort below. For orientation, we first give a brief overview of the DOOCS code base.

CODE ORGANIZATION

DOOCS consists of multiple libraries, tools, and servers, the biggest part of which is written in C++[1]. Two libraries form the core of almost every DOOCS application:

- The DOOCS client library (*clientlib*) provides the basic functionality to list `names()` from the DOOCS namespace and to send `get()` and `set()` requests over the network. It also offers interoperability with other control systems such as EPICS [10] and TINE [11, 12] by exposing some of their native functionality via the DOOCS API.
- The DOOCS server library (*serverlib*) provides the building blocks for a DOOCS server which accepts requests from the network. It contains classes for *properties* of various data types and allows instantiating these properties as members of *locations* with user-defined functionality. It also handles archiving, configuration management and related tasks.

These two libraries are complemented by approximately one hundred other DOOCS-related libraries of various purposes, ranging from support for specific hardware like cameras over interfaces to data acquisition (DAQ) systems to high-level algorithms for particle tracking.

Most of the DOOCS applications are servers. They connect hardware devices such as beam position monitors or vacuum pumps to the network, process and archive data, run feedback loops, and execute complex algorithms for advanced data evaluation. Currently, our repositories contain source code for more than 500 different server types. For most of these, multiple instances are running at one or more of DESY's accelerator facilities. Between libraries and servers, the C++ code base consists of ~8000 source files with a total of ~1.5 million lines of code[2].

Figures 1 and 2 show the development of the number of lines of code in the client- and serverlib over time. It is worth noting that the clientlib was dominated by C code between

* lars.froehlich@desy.de
† retired

[1] Notable exceptions are the JDOOCS client library for Java and tools based on it such as the graphical user interface builder *jddd* [9]. Client libraries for Python, Matlab, and LabView also exist and have spawned a multitude of tools in these languages.

[2] We count lines of code excluding comments and blank lines using the `cloc` [13] utility.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV027

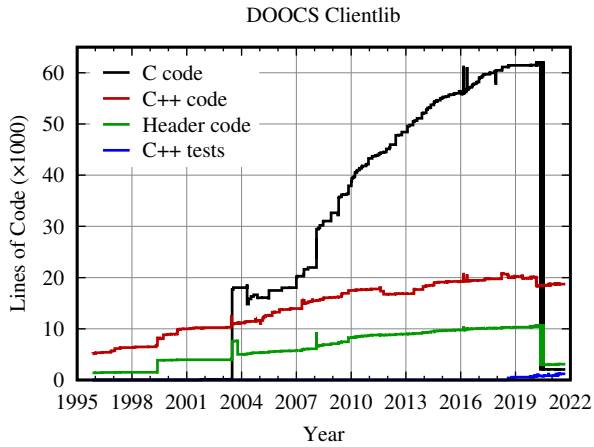


Figure 1: Lines of code excluding comments and blank lines in the DOOCS clientlib. The plot differentiates between C implementation files, C++ implementation files, header files, and C++ unit tests. From 2003 to 2020, the library was dominated by C code from the included TINE client library.

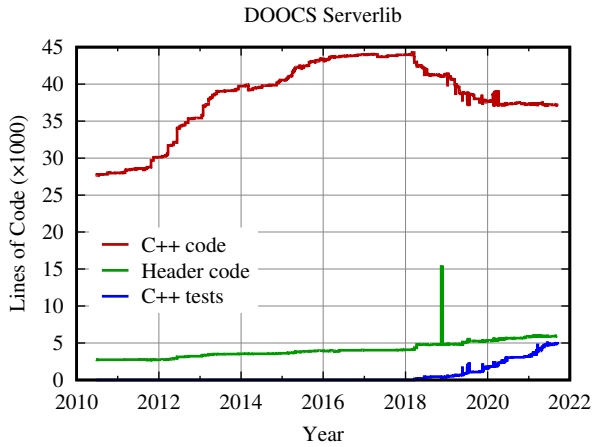


Figure 2: Lines of code excluding comments and blank lines in the DOOCS serverlib. The plot differentiates between C++ implementation files, header files, and C++ unit tests. Development started in 1992, but the history preceding the year 2010 has been lost.

2003 and 2020 because of the inclusion of the full client and server sources for the TINE control system. Since last year, TINE is linked as an external shared library. This has allowed us to preserve the full set of interoperability features without having to deal with compiler warnings from its legacy code base. A small amount of C code still remains for the interface with the RPC library.

MODERNIZATION OF CODE AND DEVELOPMENT TECHNIQUES

Modernizing a code base whose oldest parts date back to 1992 has its risks. For the most part, bugs in the existing code have been ironed out many years ago, and refactoring tends to introduce new ones. The almost complete absence of unit tests until 2018 means that inadvertent behavior changes introduced through modifications are very hard to detect. Many desirable changes in libraries are also impossible without sacrificing the stability of the application programming interface (API), causing the need to adapt huge numbers of dependent projects.

While all of these can be good reasons to leave existing code untouched, we also observed several tendencies that slowly became a cause for concern:

- Certain parts of the code had become so complex that they could only be understood by single developers.
- Introducing new features was becoming harder and more error-prone.
- Attracting new programmers to actively participate in the development was becoming harder.

All of this lead us to the conclusion that a careful modernization not only of the code base, but also of our development techniques would be necessary to prepare DOOCS for future projects. We set ourselves the following goals:

- Improve the *readability* of the code to make it easier for all developers to understand how it works.
- Improve overall *maintainability* by better code organization and reduced complexity.
- Improve *stability* by eliminating sources of bugs, memory leaks, and undefined behavior.
- Improve *teamwork* by collaborative development techniques and code reviews.

We discuss some of the measures taken in this spirit below, but the list is necessarily incomplete. In fact, outside of the core libraries individual developers have wide freedoms to adopt them only partially or to go far beyond them.

More Modern C++ Constructs

A huge part of the changes that started in the middle of the 2010s regards the use of the C++ language and its standard library itself:

Language standard: Solaris was dropped as a development platform for DOOCS in 2018. This has allowed us to use the C++14 [14] standard with the compilers for all remaining supported platforms since then. We expect to upgrade to C++17 [15] and possibly later standards in the coming years when support for a few old Linux distributions is dropped.

Templates had practically been banned from the core libraries after bad experiences with compilers in the 1990s. Through their (re-)introduction, the amount of duplicated code could be reduced dramatically while improving the uniformity of interfaces and their type safety at the same time. One of the visible effects for DOOCS users is the availability of a full set of signed and unsigned integer types in various bit widths.

Const-correctness was largely ignored in the original APIs. Retrofitting `const` modifiers across API and inheritance boundaries can be a delicate task as it often involves changes

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV027

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

to multiple projects at the same time. Although a lot of work remains to be done, the public APIs have already improved a lot and make it much easier to write const-correct code.

String handling in the original core libraries almost exclusively meant dealing with C strings or raw memory. Some of that code was rewritten to use the standard `string` or `string_view` classes, dramatically reducing complexity, line count, and room for errors. Virtually all of the user-facing API now uses these classes in function signatures instead of `const char*` or even `char*`.

Memory handling was mostly based on raw pointers and explicit use of `new` and `delete`, making new code prone to memory leaks. This type of code is gradually being replaced by STL containers, RAII (resource acquisition is initialization) types, and smart pointers.

Algorithms from the STL were not widely used in the original code base. Their increased application helps make many code passages much more succinct and readable.

Standard types have replaced entire platform-specific libraries. For instance, the standard `thread` is now used instead of its counterpart from the POSIX threads library.

Strong types are gradually being introduced to replace fundamental types like `int` for specific applications (e. g. `Timestamp`, `EventId`). This makes it harder to confuse function parameters and allows enriching the type with functionality and invariants.

Unit Tests

Apart from a few very limited integration tests, none of the core libraries had a real test suite until 2018. At this point, we started writing unit tests using the Catch2 [16] framework. Most tests are written to assert the current behavior of the code before modifying or refactoring it. Unfortunately, parts of the code have complex dependencies that make testing hard. This situation can only be improved through partial rewrites in the long run. Both the client- and the serverlib remain severely undertested with 1 line of tests for 16 lines of code for the former and 1:8 for the latter, but the test suites are growing continuously.

Build System

We are in the process of changing our build system from `make` to Meson [17]. Overall, this makes our build setup much simpler and less platform dependent. Meson was chosen over similar alternatives because of its comparatively simple syntax and because of in-house expertise.

General Utility Library GUL14

Even in comparatively small C++ code bases, some basic functionality gets implemented again and again because it is not available through the standard library – for instance, a function to determine if a string contains another string[3]. This has lead to the development of several *base libraries* [18–20] in the industry attempting to fill this gap.

[3] A `contains()` member function for the standard `string` class is only expected for the 2023 language standard.

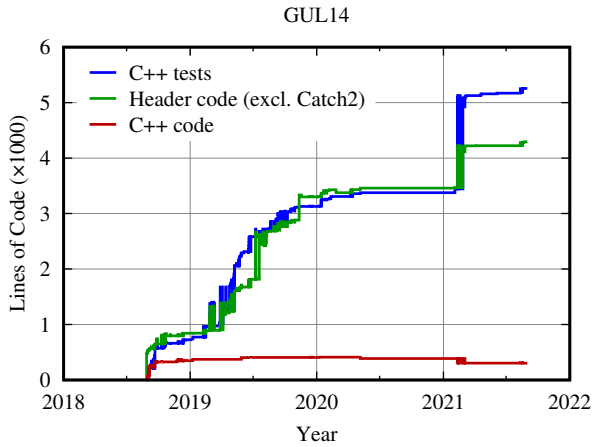


Figure 3: Lines of code excluding comments and blank lines in GUL14. The plot differentiates between header files, C++ implementation files, and unit tests. We do not count the header of the Catch2 unit testing framework which is included in the library.

We evaluated several of them and, amid concerns about their impact on our build process, found all of them lacking some specific functionality that we needed.

In August 2018, we therefore started the development of our own *General Utility Library for C++14* (GUL14). The library gathers code of wide applicability that has no external dependencies except for the C++ and C standard libraries. Specifically, it is free from control system specific code. Among other fields, it covers string operations, concurrency, time, statistics, numeric utilities, containers, and backports of features from post-C++14 standard libraries (e. g. `string_view` and `span`).

As it forms the basis for all other DOOCS libraries and applications, we strive for a very high quality level for GUL14:

- *Style:* Code must adhere to our C++ style guide and it should follow the C++ Core Guidelines.
- *Documentation:* Every function, type, or other entity in the API must be fully documented. This includes a clear description of purpose and functionality. For functions, all parameters, return values, thrown exceptions, and pre-/postconditions are described. For classes, all class invariants are clearly stated.
- *Unit tests:* Every entity in the library must have a set of associated unit tests.
- *Code review:* Every commit to the library must be reviewed by at least one developer. Every developer must ensure that all of the other quality criteria are fulfilled.

As shown in Fig. 3, the library is dominated by template code residing in header files. The unit test suite has more lines of code than the library itself.

GUL14 is open source and published under the GNU Lesser General Public License v2.1 [21]. Unfortunately we currently cannot publish the code on a third-party platform

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV027

like Github, but the source code and documentation are publicly available [22] and external contributions are welcome.

Code Review

Code reviews are an excellent method to screen code changes for defects. Maybe more importantly, they foster mutual learning between developers and thereby help spread expert knowledge. Unfortunately, they also take a substantial amount of time. Owing to the already high workload on our development team, we have adopted a multi-layered guideline:

1. For GUL14, every commit must be reviewed and accepted by at least one other developer.
2. For the client- and serverlib, all but trivial commits must pass review.
3. For other libraries, code review is strongly encouraged.
4. For servers and tools, code review is optional.

Code reviews typically take place via merge requests on an on-site Gitlab web server.

Continuous Integration

The code for the core libraries is automatically compiled and tested on multiple platforms by the Gitlab system after each commit. In part, this includes address sanitizer and undefined behavior sanitizer builds, which generally provides excellent early warnings against faulty code. We are also looking into continuous delivery (CD) workflows for the future.

Cross-Project Refactoring

Some refactorings, especially API changes in libraries, require invasive modifications in user code. Although we generally try to avoid such changes, a few of them are deemed essential to fix architectural problems. Since DOOCS is used almost exclusively at DESY with only few external users, we are in the lucky situation that we have control over almost all user code in our version control repositories. Refactorings of low complexity (such as renaming a class member or a header file) can therefore be automatically applied via text substitution scripts. Where possible, the old API is marked as deprecated and remains available for 1–2 years until it is finally removed.

Training

Educating new programmers is essential to maintain a capable control system software team. To this end, we are providing a regular series of remote lectures on DOOCS related topics. A C++ style guide provides orientation on common coding conventions.

CONCLUSION

Next year, DOOCS will turn 30. As in any code base of this age, some problems have accumulated over time and tend to hinder further development. Since the mid-2010s, however, we have invested more and more effort into the continuous refactoring of the code and into a modernization of our development process. We try to take advantage of established best practices, of recent improvements of the C++ language itself, and of modern tooling to improve the quality of the code and the cooperation between the developers. The feedback received from our users makes us hopeful that we are on the right path to prepare DOOCS for the future.

ACKNOWLEDGEMENTS

We would like to thank J. Georg, M. Hierholzer, D. Kalantaryan, M. Killenberg, and T. Kozak for contributing patches and bug reports for the core DOOCS libraries. Innumerable colleagues from the the M-, FS-, and FH- groups have helped shape the development of DOOCS over the years and deserve our gratitude. We also acknowledge the continuous support by the machine and research divisions at DESY, the Helmholtz Association of German Research Centers, and the European X-Ray Free-Electron Laser Facility GmbH.

REFERENCES

[1] O. Hensler and K. Rehlich, "DOOCS: A distributed object oriented control system", in *Proc. XV Workshop on Charged Particle Accelerators*, Protvino, Russia, 1996.

[2] DOOCS homepage, https://doocs.desy.de

[3] G. A. Voss and B. H. Wiik, "The electron-proton collider HERA", *Annual Review of Nuclear and Particle Science*, vol. 44.1, pp. 413-452, 1994.

[4] R. Srinivasan, *RPC: Remote procedure call protocol specification version 2*, IETF Request for Comments 1831, August 1995, https://www.ietf.org/rfc/rfc1831.txt.

[5] D. Bank, *The Java saga*, Wired 3.12, Dec. 1995.

[6] *International standard ISO/IEC 14882:1998*, International Organization for Standardization, Geneva, Switzerland, Sept. 1998.

[7] M. A. Ellis and B. Stroustrup, *The Annotated C++ Reference Manual*, Addison Wesley, ISBN 0-201-51459-1, May 1990.

[8] *International standard ISO/IEC 14882:2011*, International Organization for Standardization, Geneva, Switzerland, Sept. 2011.

[9] E. Sombrowski et al., " 'JDDD': A Java DOOCS data display for the XFEL", in *Proc. ICALEPCS07*, Oct. 2007, pp. 43–45.

[10] L. R. Dalesio et al., "EPICS architecture", *Proc. ICALEPCS91*, Nov. 1991, pp. 278–282.

[11] P. Bartkiewicz and P. Duval, "TINE as an accelerator control system at DESY", *Meas. Sci. Technol.*, vol. 18, pp. 2379–2386, 2007. doi:10.1088/0957-0233/18/8/012

[12] P. Duval *et al.*, "Control system interoperability, an extreme case: Merging DOOCS and TINE", in *Proc. PCaPAC2012*, Dec. 2012, pp. 115–117.

[13] cloc tool for counting lines of code, https://github.com/AlDanial/cloc

[14] *International standard ISO/IEC 14882:2014*, International Organization for Standardization, Geneva, Switzerland, Dec. 2014.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV027

[15] *International standard ISO/IEC 14882:2017*, International Organization for Standardization, Geneva, Switzerland, Dec. 2017.

[16] Catch2 unit test framework, https://github.com/catchorg/Catch2

[17] Meson build system, https://mesonbuild.com/

[18] Google Abseil library, https://abseil.io/.

[19] Bloomberg BDE libraries, https://github.com/bloomberg/bde

[20] Facebook folly library, https://github.com/facebook/folly

[21] GNU Lesser General Public License version 2.1, https://www.gnu.org/licenses/old-licenses/lgpl-2.1

[22] General Utility Library for C++14, https://winweb.desy.de/mcs/docs/gul/index.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV030

APPLICATION OF EPICS SOFTWARE IN LINEAR ACCELERATOR

Yuhui Guo†, Nian Xie, Rui Wang, Haitao Liu, Baojia Wang
Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou, China

Abstract

The institute of modern physics (IMP) has two sets of linear accelerator facilities, they are CAFe (China ADS front-end demo linac) and LEAF (Low Energy Accelerator Facility). The Main equipment of LEAF facility consists of ion source, LEBT (Low Energy Beam Transport), RFQ (Radio Frequency Quadrupole) and some experiment terminals. Compare with LEAF, CAFe equipment has more and adds MEBT (Medium Energy Beam Transport) and four sets of superconducting cavity strings at the back end of RFQ. In the process of commissioning and running linac equipment, The EPICS Archiver application and Alarm system are used. According to the refined control requirements of the facility sites, we have completed the software upgrade and deployment of the archiver and alarm systems. The upgraded software system have made the operation of linac machines more effective in term of monitoring, fault-diagnostic and system recovery, and becomes more user-friendly as well.

INTRODUCTION

The CAFe is a prototype of ADS proton superconducting linear accelerator. It mainly includes a compact ECR (Electron Cyclotron Resonance) ion source, a low energy beam transport LEBT section, a radio frequency quadrupole acceleration system RFQ , a MEBT, 4 sets of CM superconducting cavity strings, a HEBT (High Energy Beam Transport) and a beam dump with the function of measuring current intensity and absorbing the energy from ion beam [1,2]. The layout of CAFe facility is shown in Fig. 1.

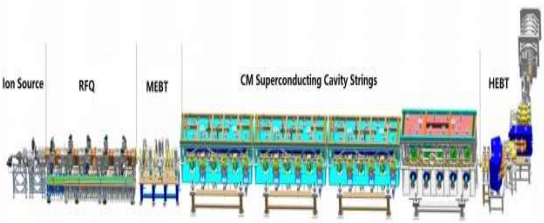


Figure 1: The layout of the CAFe facility.

The CAFe facility can provide a high power beam with the beam intensity of several mA. At the same time, CAFe, as a new research device, needs to do some research on the operation mode of the machine.

The LEAF contains a 45 GHz fourth-generation superconducting high-charge state ECR ion source, a 300 kV high-voltage platform, an advanced RFQ heavy ion accelerator capable of accelerating a variety of ions, and a DTL energy regulator. LEAF is a research facility of the low-energy, high-current and high-charged heavy ion [3]. The layout of the LEAF is shown in Fig. 2.

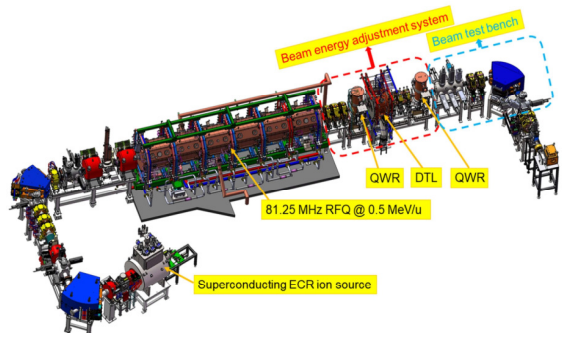


Figure 2: The layout of the LEAF facility.

LEAF can provide pulse (beam energy range of 0.1-20 MeV) and continuous beam with high beam intensity, high charge state and many kinds of ions. The core parts of the LEAF are superconducting high charge ECR ion source and RFQ accelerator. They are the most key equipment of the front-end injector for the next generation large heavy ion accelerator. Therefore, the development of the LEAF will build a good foundation for the development of the next generation high-power high current heavy ion accelerator.

During the commissioning and operation of the LEAF and CAFE accelerator systems, the upper-level applications have a similar requirement for both systems. For example, the alarm system needs to monitor whether values of the defined PV variables exceed their safety threshold ranges, and the data archiving system needs to store the historical relevant PV data from the systems for fault location diagnosis and error analysis.

For the alarm system, the BEAST software package is used for the LEAF, which is mainly composed of Alarm-ConfigTool, Alarm Server, JMS (Java Message Server) and Alarm Client GUI. In the control system of the CAFe facility, we use the alarm module from the EPICS Phoebus software to upgrade the BEAST-based alarm system. Compared with ActiveMQ message publishing technology, Kafka with its own alarm message publishing technology has much higher data throughput and is more suitable for processing large-scale real-time data streams.

For the data archiving system, we used the Channel Archiver and Archiver Appliance software packages in the EPICS software tool to build two data archiving systems with redundant functions. The Channel Archiver software package was released earlier, and its client software is relatively rich. The Archiver Appliance is a new data archiving system and adopts a multi-level storage method. It also provides a Web front-end management interface, allowing

†Email address: guoyuhui@impcas.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV030

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

users to directly archive, query and mange PV variables through web pages.

OVERALL STRUCTURE OF THE SOFTWARE SYSTEM

As the types of field equipment of linear accelerator are varied and the installation positions are relatively scattered, we use a network-based distributed software structure to construct the upper-level application software system. The software structure is shown in Fig. 3.

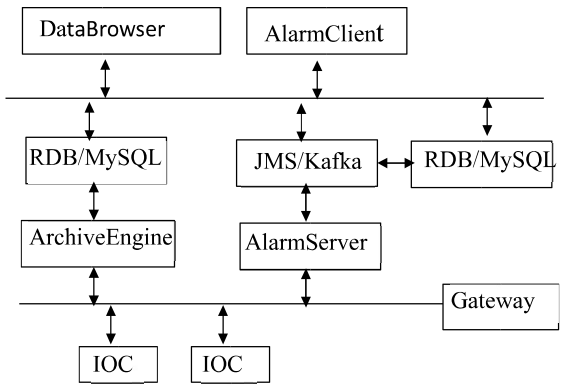


Figure 3: Software structure of data archiving and alarm system.

The CSS-based DataBrowser and AlarmClient tools are used for our operation interface which provides the functionalities of querying the operating status of the equipment and displaying the alarm information. For the alarm system and archiver system, they have been very well integrated in CSS. By relatively simple parameter configurations, Users can easily query and display required operation data.

ArchiverEngine is mainly responsible for reading the real-time data from listed PVs from all IOCs, and then saving it to the historical database; AlarmServer is the core of the alarm system and communicates with the IOC through the CA (Channel Access) protocol to monitor the alarm status of PV values and publish the alarm messages.

In order to improve the reliability of the data link connection when reading a large number of PV variables, and at the same time to classify and manage these data, the EPCIS gateway tool is used between IOCs and the upper data processing engine to achieve the security control of data access and data classification.

ALARM SYSTEM

When the operating status of the field equipment changes, especially when a fault occurs, the alarm system will issue alarm information to the accelerator commissioning, operation and maintenance personnel or equipment maintenance personnel through software user interface, voice, SMS(text messages) or other methods. The alarm system should not be used to replace the equipment protection functions of the Interlock system. It provides data for equipment operators to analyze the causes of equipment failures, but sometimes it also increases the burden of data analysis for operation and maintenance personnel. During accelerator beam commissioning, the physical personnel can use the electronic log software to realize the recording function of the alarmed equipment and the processing methods of the operation and maintenance personnel. Clog electronic log is used in CAFE facility, which is provided by CSNS.

The alarm system based on the BEAST software package can import the database configuration file into the relational database through the AlarmConfigTool tool. The Alarm Server is used to read the configuration information from the relational database, monitor the alarm information of the PV channels in the IOCs, and pass the alarm information to the client through the JMS; JMS acts as a communication channel between alarm client and server to transmit alarm messages, receive log information of other applications etc. The Alarm Client GUI is used to display alarm information in real time. The demand statistics of some alarm equipment of LEAF system is shown in Table 1.

Table 1: Main Alarm Device Information Statistics of LEAF Facility

Device name or system	Threshold Value	Alarm message is displayed
RFQ cooling water system	Water temperature 36°	RFQ water temperature alarm
	Water flow 20L/min	RFQ water flow alarm
	Water pressure 0.3Pa	RFQ water pressure alarm
Vacuum system	LEBT 10-5mbar	LEBT vacuum alarm
	RFQ 1.1x10-5mbar	RFQ vacuum alarm
Water quality monitoring	Water resistance 9 MΩ	Water resistance alarm
	Level of water tank 1.5m	Water level alarm of tank
	Supply-water pressure 4 bar	Water pressure alarm of water

The data source monitored by the Alarm system can come from the device IOC or the PV variables published by the soft IOC. For example, the Alarm system of CAFe can monitor the change of the PV values published by using the pcaspy software package to realize the alarm of the soft protection logic value in the MPS system. The operation interface of the alarm system is shown in Fig. 4.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV030

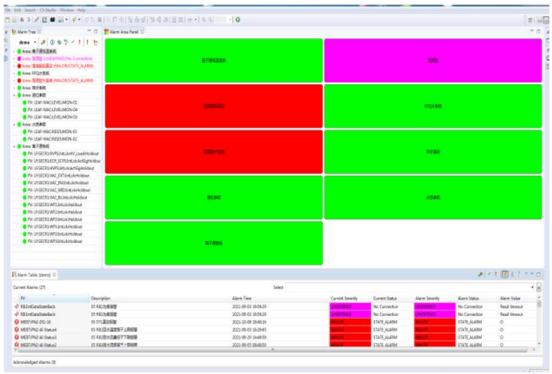


Figure 4: Operation interface of the Alarm system.

As shown in Fig. 4, the alarm system of LEAF includes 9 alarm modules, which are ion source cryogenic system, vacuum system of energy increasing and decreasing section, radio -frequency cavity water temperature system, liquid level parameters, ion source system, radio-frequency cavity, RFQ Water system, beam diagnosis system and water quality system.

In the construction of the alarm system of the CAFe facility, we adopt the Phoebus-based CSS software system for interface development and deployment. Phoebus is the current variant of Control System Studio [4]. Compared to the environment of the early CSS software, it no longer depends on EclipseRCP. Instead, it offers software tools such as Display Builder, Data Browser, Probe, PV Tree, Alarm, Scan and so on. The Phoebus software architecture is simpler, and the development environment is easy to set up, all of which help greatly improve the user experience. It is a very suitable for the development of control system operation interface.

Alarms in Phoebus are composed of server and client modules [5]. The server sends alarm messages to the Kafka middle message plug-in by monitoring the changes in the alarm status of the PV variables issued by each IOC. The alarm interfaces that can be displayed on the Phoebus software are Alarm Tree, Alarm Table and Alarm Area Panel.

The alarm module in Phoebus software is used to deploy the alarm system, which can provide users with a concise, manageable and operable alarm interface, realize the visualization of alarm data and improve the real-time performance of fault handling.

DATA ARCHIVER SYSTEM

In the process of accelerator commissioning and operation, the equipment fault location and fault analysis based on the data archiving system is very important. When equipment failures or beam current accidents occur, accelerator physics personnel need the data archiving system to provide historical data of related equipment to facilitate post-accident handling and cause analysis.

Previous archive systems were built based on the EPICS Channel Archiver tool, using MySQL for data storage and running a database server on Linux, and mounting the IBM DS3512 disk array for data storage as well. KeepAlived is used to achieve high availability of the database server [6]. The archive system communicates with the front-end IOC using the ArchiveEngine tool, and the browsing and query of archived data is realized by the DataBrower tool in the CS-Studio software.

As the scale of data in the archive database becomes larger and larger, the speed of reading and retrieving the data stored in the MySQL database becomes slower and slower. At the same time, the BEAUTY-based archive system does not provide friendly interfaces of information query and management. Therefore, we upgraded the archive system of these two facilities (LEAF and CAFe) with Archiver Appliance.

The new archiving system includes IOC data collection, data archiving engine (Archiver Appliance) and data query and information management tools. Its software system structure is shown as in Fig. 5.

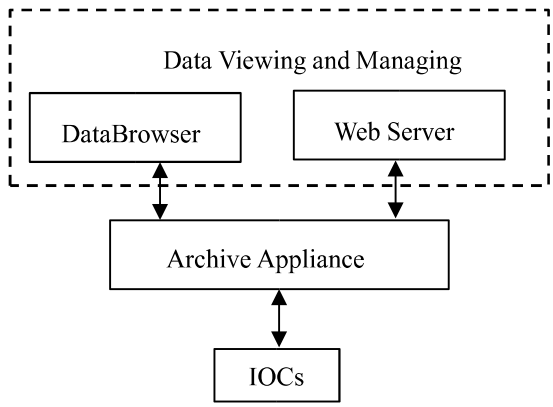


Figure 5: The structure diagram of Archiver Appliance.

The Archiver Appliance reads PV data according to the PV variable name, data sampling mode and frequency in the configuration file, and stores the collected data in a structured file in PB format. The DataBrowser supports Archiver Appliance's PB/HTTP protocol and accesses it through the pbraw interface. Users can directly access and query the Archiver Appliance data system through a browser and receive Web response of user filing to their query requests. The web operation interface of the Archiver Appliance system is shown in Fig. 6.

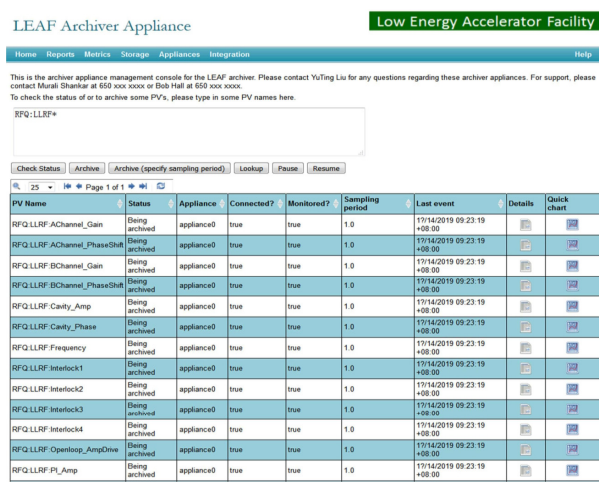


Figure 6: The Web operation interface of Archiver Appliance.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV030

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The data archiving system based on the Archiver Appliance solves the problems of low data index efficiency and inconvenient background management in the past. At the same time, the upper human-computer interaction interface is designed by using CSS software system based on Phoebus, which provides convenience for physical personnel to access experimental data and fault diagnosis and analysis.

CONCLUSION

Compared with the alarm system of ActiveMQ message mechanism, the accelerator alarm system based on Kafka streaming data platform can effectively improve the data throughput of the system and the real-time performance of alarm information release.

The archive system software of Channel Archiver was released earlier, and its client software is relatively rich. The new Archiver Appliance software uses a multi-level storage method, which can retrieve 1 Hz double-precision data for one day within 0.5 seconds. It also provides a Web front-end management interface, which has more advantages in system information management and fast data retrieval.

After the upgrade and redeployment of the above application software, the usability of the data service in the control systems of the two facilities is improved, and better user-friendly interfaces are provided for data query and operations. The upgraded software systems run more stable, and the user interfaces are more friendly, which improves the maintainability of the accelerator device for long-term operation.

REFERENCES

[1] Yuhui Guo, Haitao Liu, Jing Wang *et al*., "Progress of the Constrol Systems for the ADS Injector II", in *Proceedings of ICALEPCS2015*, Melbourne, Australia, WEPGF011.

[2] Yi Cheng, Hai Zheng, Haitao Liu *et al*., "Design of machine protection system for cafe facility", *Radiation Detection Technology and Methods*, DOI 10.1007/s41605-020-00196-8, (2020).

[3] Yuhui Guo, Nian Xie, Zhangnuo Chen *et al*., "Design of LEAF control system", *Radiation Detection Technology and Methods*, DOI 10.1007/s41605-019-0137-8, (2019).

[4] SNS Controls CS-Studio – Oak Ridge National Laboratory, https://controlssoftware.sns.ornl.gov/css_phoebus/.

[5] Alarms-Phoebus 1.0 documentation, https://control-system-studio.readthedocs.io/en/latest/app/alarm/ui/doc/index.html.

[6] Jiangbo Luo, Yuhui Guo, Haitao Liu *et al*., "Data archiving system for injectorII of accelerator driven sub-critical system", *High Power Laser and Particle beams*, pp.128-133, Chinese (2016).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV031

THE DEPLOYMENT TECHNOLOGY OF EPICS APPLICATION SOFTWARE BASED ON DOCKER

R. Wang†, Y. H. Guo, B. J. Wang, N. Xie, IMP, Lanzhou 730000, P. R. China

Abstract

StreamDevice, as a general-purpose string interface device's Epics driver, has been widely used in the control of devices with network and serial ports in CAFe equipment. For example, the remote control of magnet power supply, vacuum gauges and various vacuum valves or pumps, as well as the information reading and control of Gauss meter equipment used in magnetic field measurement. In the process of on-site software development, we found that various errors are caused during the deployment of StreamDevice about the dependence on software environment and library functions, which because of different operating system environments and EPICS tool software versions. This makes StreamDevice deployment very time-consuming and labor-intensive. To ensure that StreamDevice works in a unified environment and can be deployed and migrated efficiently, the Docker container technology is used to encapsulate its software and its application environment. Images will be uploaded to an Aliyun private library to facilitate software developers to download and use. At the same time, some technical implementations, to ensure the safety of containers and host, for communication security between containers and system resource management should be taken on our host. In the future, the large-scale efficient and stable deployment of StreamDevice software can be realized by pulling images on the server installed with Docker, avoiding a lot of repetitive work and greatly saving the time of control technicians.

INTRODUCTION

CAFe equipment, the CiADS superconducting linear accelerator prototype, was built in 2018 to validate the superconducting linear accelerator for CiADS with 10 mA continuous beam current. The equipment consists of the ion source, low-energy transmission section, RFQ, medium-energy transmission section, superconducting linear accelerator, high-energy transmission section, and beam terminal collector. The equipment layout is shown in Fig. 1. The overall parameters of the equipment include the beam intensity of 10mA, the beam energy of 25-40MeV, the RF frequency of 162.5MHz, as well as the temperature of 4.5k [1-3].

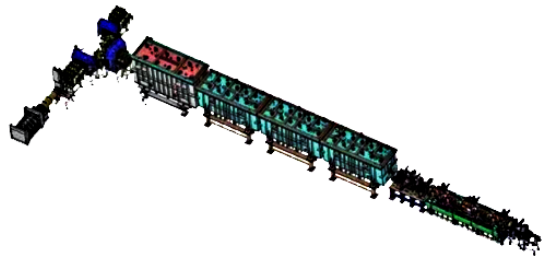


Figure 1: Layout of CAFe superconducting proton linear accelerator.

At present, EPICS is mostly used in the CAFe equipment control software of the Institute of Modern Physics, Chinese Academy of Sciences. The experimental physics and industrial control software adopt the standard distributed control system model and run stably. The device driver is developed with unified system architecture. It has gradually replaced LabVIEW as the mainstream software architecture and development tool for the accelerator control system in China and abroad. StreamDevice module is needed to realize serial communication equipment and drive control, which has the interface of the asynchronous drive module (ASYN) and can provide equipment support for EPICS [4].

With the upgrading of CAFe equipment, the servers have also been updated. In practice, it is found that there are often encountered unknown errors in the installation of EPICS on each new server. A lot of manpower and time will be paid to find the cause of the problems and solve methods. But the solutions do not possess universality. In order to improve the deployment efficiency and reduce the difficulty of application migration, Docker [5] is used to encapsulate EPICS + StreamDevice to assure the unification of the operating environment on different machines. At the same time, a series of security problems existing in the current container should be optimized and improved.

FUNCTIONALITY AND REQUIREMENTS

An asynchronous drive module (ASYN) interface was integrated with StreamDevice, which supports serial ports (RS485, RS232), IEEE-488 (GPIB or HP-IB), and Ethernet interfaces. It can realize the communication between serial ports and Gaussian power supply. The communication rules are specified by writing protocol files, and PV information is defined by DB files [6]. The result of the EPICS reads and sets the equipment information was achieved. Because the StreamDevice, a model of EPICS, was used to develop of I/O IOC for power and Gauss meter.

StreamDevice can analyze web pages by regular expressions, as well as read this important information in real

† wangrui@impcas.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

time in the web control interface to learn whether the archiving system is running. After setting PV, CA Server performs data archiving by Archive Engine. And these data should be submitted by Achieve Export to browse. Finally, the information is displayed in the human-machine interface on Web Apps [7]. The process is shown in Fig. 2.

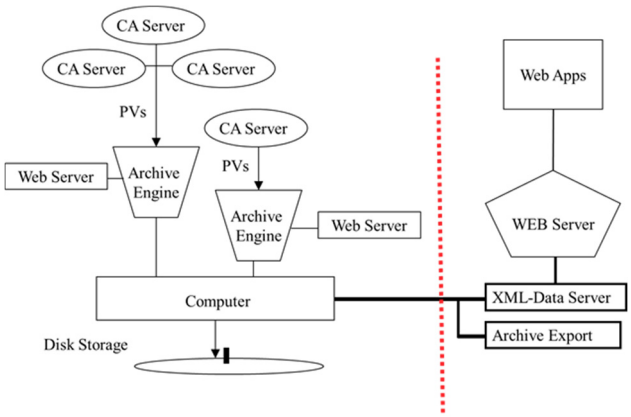


Figure 2: StreamDevice is used to monitor the running status of your app and parse web pages.

INTEGRATION OVERVIEW

EPICS has been widely used in the field of CAFe equipment control. The application based on EPICS has good portability, scalability, reusability, and maintainability. EPICS can be divided into three parts from top to bottom: the upper layer is the operator interface OPI, which can complete the human-machine interaction (HMI) process. The middle layer is the input and output controller, referred to as the IOC. The lower layer is channel protocol CA, CA protocol is divided into two parts: CA client and CA server [8]. The servers running IOC can be accessed via TCP/IP by the client. ASYN provides driver support for communication with hardware devices. StreamDevice, a generic EPICS device, can control devices by sending and receiving strings [9].

Docker, a lightweight application container deployment framework, can package, publish, and run any applications [10]. Application migration can be realized by firstly packaging EPICS+StreamDevice with Docker, secondly building images and sending them to the remote registry (docker hub), finally pulling and running images on deployed servers. The overall framework of the system is shown in Fig. 3.

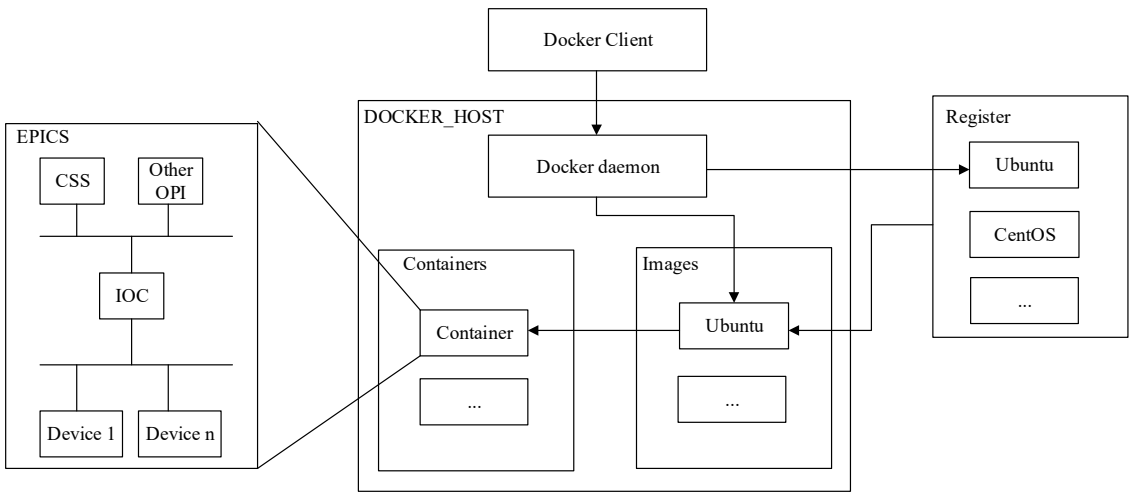


Figure 3: The overall framework of the system.

DOCKER CONTAINER TECHNOLOGY AND SECURITY

Docker

Docker is considered as an Internet container and will be the future standard for PaaS [11]. Before using Docker to package EPICS + StreamDevice, if the server is updated or the server is abnormally collapsed, these applications need to be redeployed. The deployment process is complex and may encounter unknown problems, which takes a lot of manpower and time. Packaging all effective loads through Docker enables consensus migration between almost any server, which greatly simplifies deployment and maintenance. The official Ubuntu 18.04 is used as the basic image. The container should be constructed from the image, which installs EPICS + StreamDevice in the container. After all, the container will be committed as a new image.

Images Security Restrictions

The official image registry allows all Docker hub users to visit. All the application projects on it are open source, which allows all users to pull and change. This openness breaks the technical barrier and facilitates project delivery. At the same time, there are some immeasurable security risks of data breaches and image misappropriating [12]. The private image stowage was built through Aliyun to increase the authentication mechanism when attempting to pull the image. This method can limit the pull permissions, as well as ensure the security of the image.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV031

Container Resource Security Mechanism

The container is different from the virtual machine. There is no independent resource allocation to make it impossible to isolate resources at the system kernel level. Since the Docker container shares the operating system kernel with the host, there is a risk that the container is controlled by the attacker to obtain the access rights of the host file. Or the attacker obtains the root permission by illegal means to control the host and other containers on the host. It will result in the escape of the container, or even that the Docker container exhausts the host to make the host or other containers pause or die [13]. In summary, it is necessary to limit the resources of container access to host through Cgroup and SELinux, and the details will be described in the test section.

Container Network Security Mechanism

Docker's default network communication mode is bridge mode, which will virtualize a docker0 bridge on the host computer, and connect all the containers used in this mode on the host computer to the bridge. At this time, all Docker containers can reach each other, which makes attackers may threaten the safety of the host computer and the container through broadcast storms, sniffing, ARP fraud, and other attacks [14]. When Docker containers provide services alone (such as EPICS + StreamDevice in this paper) without requiring multiple Docker containers to form micro-services, it is necessary to prohibit communication between containers, which can be achieved by setting the parameter dockerd-icc.

PERFORMANCE

Image Testing

The basic image used in this paper is the official image of Ubuntu 18.04. Based on this image, the dockerfile [15] file is written. The basic image, the author's signature, the working path, and the container number are specified. The host directory file is mounted and the environmental path is edited. Each step can be seen as an additional layer in the original image.

After entering the container, EPICS is installed and then Asyn and StreamDevice are deployed. The deployment results and PV display results are tested. EPICS + StreamDevice in Fig.4 has been successfully deployed in the container. Committed the successfully tested container as a new image and pull the packaged image on the required server.

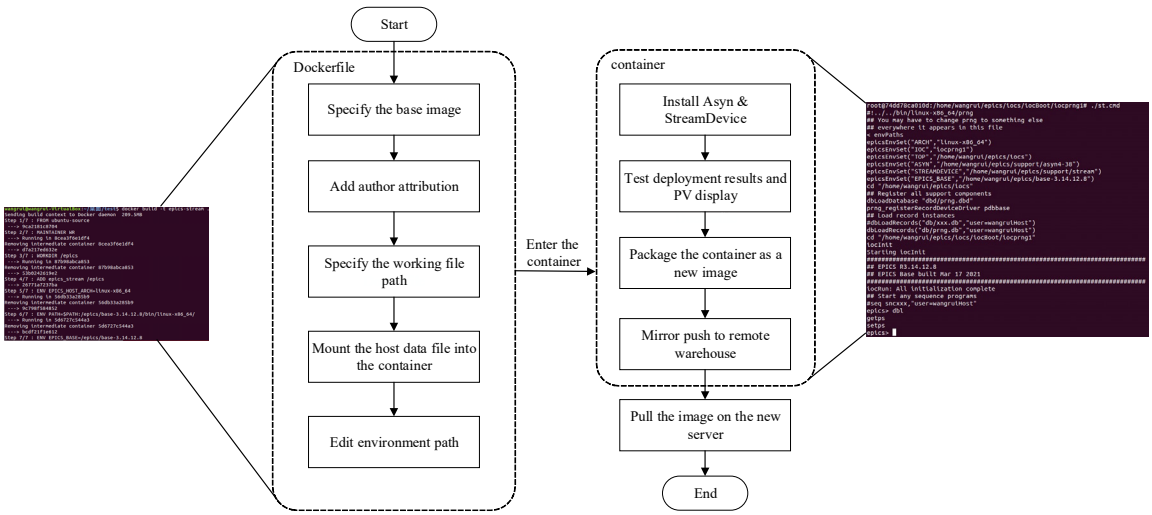


Figure 4: Containers are built by dockerfile and EPICS was deployed successfully with the PV is displayed.

Mirror Access Authentication Test

By establishing a private image stowage under the Aliyun platform, the problem of image pulling authentication can be solved. First Aliyun account should be logged in to establish an image stowage and create a namespace. Then the image Tag that needs to be uploaded should be changed on the host running docker. The image is uploaded through the push command. When the image needs to be pulled, the remote Aliyun image stowage needs to be logged in through docker, as well as the pull command can be used only after the authentication is completed, which limits the pull of private image by users who cannot complete the login authentication to a certain extent. The process is shown in Fig. 5.

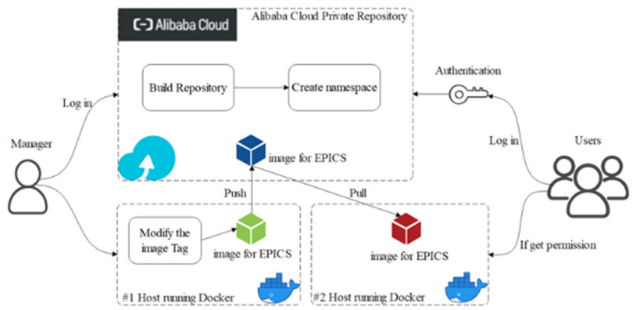


Figure 5: Aliyun private repository image access authentication.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Resource Limit Testing

Cgroups can limit resource items such as CPU, memory, and disk I/O speed of the Docker container to avoid a single container running out of hardware resources on the host. Table 1 takes memory as an example to show the difference of system resources occupied by containers under the condition of whether to open Cgroups constraints. After the Cgroups memory limit was opened, the process is killed if the container actually uses more memory than the limit. If Cgroups was not used to limit memory, the container will occupy too much memory of the host, which may lead to the death of other containers. Experiments show that the container takes up five times more memory than using Cgroups. Finally, Docker's mandatory access control of host resources is implemented by starting the SELinux mechanism.

Table 1: Cgroups Memory Limit Comparison

Status	System total memory(MB)	Set limited memory(MB)	Memory usage(MB)	Percentage of total memory
Use Cgroups	2048	200	1.08	0.05%
Do not use Cgroups		—	5.14	0.25%

Network Communication Testing

The EPICS + StreamDevice container used in this paper provides services separately, so it should be forbidden to communicate with other containers running on the host computer. In the test environment, the IP of the EPICS + StreamDevice container is 172.17.0.2, and that of the Ubuntu container is 172.17.0.3. It is necessary to set dockerd-icc = false to prohibit communication between them, as shown in Fig. 6.

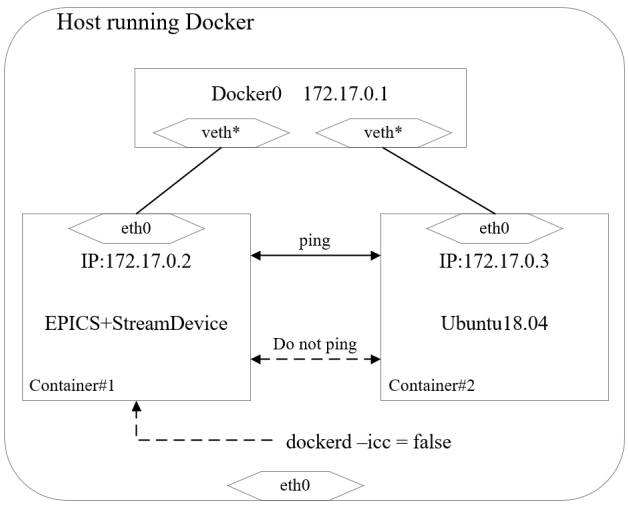


Figure 6: All containers running on the same host are in the same network segment, and if they are not restricted they can ping through each other, restarting the Docker service by limiting the icc parameter can achieve the purpose of prohibiting communication between two unrelated containers.

SUMMARY

The packaging of EPICS application by Docker can effectively reduce the deployment difficulty of EPICS. It also reduces the cost of application migration and solves a series of problems caused by the difference between the test environment and the production environment. At the same time, a series of improvements are made on the level of container for data information security problems in a production environment. The problem of container escape and container exhaustion of host resources is effectively prevented by container resource constraints of Cgroups and mandatory access control of SELinux. The container network security issues should be replied by prohibiting communication between containers that do not form a microservice, which can effectively reduce the risk that attackers misappropriate the information security of other containers or even host computers by holding a container. Finally, by uploading the fabricated image to the private cloud platform, the authentication mechanism is added to ensure the application of the image and its internal storage, as well as the safety of the data. The application of the accelerator control system can be deployed efficiently by the combination of Docker and EPICS technology, which can also save manpower and time. In this way, this technology has broad application prospects in the field of accelerator control.

REFERENCES

[1] Y. H. Guo *et al.*, "Progress of the Control Systems for the ADS injector II", in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, pp. 709-712. doi:10.18429/JACoW-ICALEPCS2015-WEPGF011

[2] T. Birke, D. Faulbaum, M. G. Giacchini, D. Herrendörfer, P. Stange, and R. Lange, "BESSY Control System Adminis tration and Analysis Tools", in Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07), Oak Ridge, TN, USA, Oct. 2007, paper RPPB28, pp. 671-673.

[3] CHEN Zhang-Nuo, "Development and design of upper application software for CAFe control system", 2020, University of Chinese Academy of Sciences (Institute of Modern Physics, Chinese Academy of Sciences) (In Chinese).

[4] Ranchal, Rohit, *et al.*, "Epics: A framework for enforcing s ecurity policies in composite web services," IEEE Transact ions on Services Computing 12.3 (2018): 415-428.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV031

[5] Zhang, Tong and Dylan Maxwell, "Cloud Computing Platform for High-level Physics Applications Development," presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOPHA167.

[6] LI Jiao-Sai, "Design of ECR ion source emittance measurement system", 2017, University of Chinese Academy of Sciences (Institute of Modern Physics, Chinese Academy of Sciences) (In Chinese).

[7] StreamDevice, https://paulscherrerinstitute.github.io/StreamDevice/

[8] EPICS IOC, https://epics.anl.gov/base/R3-14/12-docs/App Dev Guide/node4.html

[9] Qiao, Xianjie and Gang Li, "Research of streamdevice application based on EPICS", Nuclear Electronics and Detection Technology 31.10 (2011): 1073-1076.

[10] Anderson, Charles, "Docker [software engineering]", Ieee Software 32.3 (2015): 102-c3.

[11] Boettiger, Carl, "An introduction to Docker for reproducible research", ACM SIGOPS Operating Systems Review 49.1 (2015): 71-79.

[12] Bui, Thanh, "Analysis of docker security," arXiv preprint arXiv:1501.02967 (2015).

[13] REN Lan-Fang, Zhuang Xiao-Jun, FU Jun, "Research on Docker Container Security Protection Technology", Telecommunications Engineering Technology and Standardization (2020) (In Chinese).

[14] BI An Safety team. Docker Container Safety Analysis [EB\OL], https://www.frebuf.com/artcles/system/221319.html

[15] Docker compose YAML file, https://docs.docker.com/compose

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

DESIGN OF A COMPONENT-ORIENTED DISTRIBUTED DATA INTEGRATION MODEL

Z. G. Ni†, L Li, J. Luo, J. Liu, X. W. Zhou, Institute of Computer Application, China Academy of Engineering Physics, Mianyang City, China

Abstract

The control system of large scientific facilities is composed of several heterogeneous control systems. As time goes by, the facilities need to be continuously upgraded and the control system also needs to be upgraded. This is a challenge for the integration of complex and large-scale heterogeneous systems. This article describes the design of a data integration model based on component technology, software middleware (The Apache Thrift*) and real-time database. The realization of this model shields the relevant details of the software middleware, encapsulates the remote data acquisition as a local function operation, realizes the combination of data and complex calculations through scripts, and can be assembled into new components.

INTRODUCTION

Large scientific experimental devices generally consist of dozens of heterogeneous systems, each of which is also an independent control system. The control system architecture of a typical large scientific facilities is a two-tier architecture consisting of a monitoring layer of the network structure and a control layer of the fieldbus structure. The monitoring layer is deployed on the virtual server and the console computer to provide centralized operations for control, status, and data storage. The control layer is deployed on the virtual server or embedded controller to provide real-time collection and control of the device.

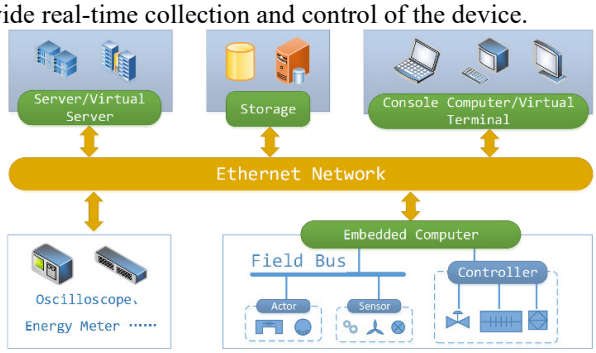


Figure 1: The control system architecture of the typical large scientific facilities [1].

As provided in Fig. 1, the monitoring layer is a software system based on Ethernet structure. It consists of a network switching system, a server system, and a console computer. It provides system services and human-machine interfaces for the facility control system, including control, monitoring, and data management.

At present, the integrated control system mainly used by large-scale laboratories and scientific research institutions around the world is EPICS or TANGO. In the automation industry, excellent instrument manufacturers will not only provide corresponding secondary development libraries, but also corresponding EPICS interfaces or TANGO Devices.

EPICS[2] is a set of software tools and applications which provide a software infrastructure for use in building distributed control systems to operate devices such as Particle Accelerators, Large Experiments and major Telescopes. Such distributed control systems typically comprise tens or even hundreds of computers, networked together to allow communication between them and to provide control and feedback of the various parts of the device from a central control room, or even remotely over the internet.

Tango[3] is an Open Source solution for SCADA and DCS. Open Source means you get all the source code under an Open Source free licence (LGPL and GPL). Supervisory Control and Data Acquisition (SCADA) systems are typically industrial type systems using standard hardware. Distributed Control Systems (DCS) are more flexible control systems used in more complex environments. Sardana is a good example of a Tango based Beamline SCADA.

Of course, many earlier control system software was built using CORBA software middleware, or implemented using the TCP custom protocol.

QUESTION

A few years later, the scientific experimental device will inevitably bring about the continuous upgrading of the system. How to adapt the control system and software to this change is a great challenge. The control system software constructed first needs to have a certain degree of scalability. Secondly, as technology evolves, the control system software needs to have a certain degree of compatibility; the early and current systems need to communicate and communicate, how to realize the communication between the early and current systems is a technical problem, which is a problem that this article needs to solve.

Another feature of the scientific experimental device is that the operator's demand is unstable. Users will constantly adjust their requirements based on their experience. We have to modify the code, compile, and test run, and the debugging time for software developers is very short, so the skills of software technicians are very high. If you can adapt to changes in requirements with zero programming, this is a great thing.

Therefore, we propose a component-oriented distributed data integration model, which is used to construct an implementation method of data acquisition and control from the device to the user interface.

In the following chapters, we first introduce the technologies that may be involved.

† email address: drops.ni@caep.cn.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

RELATED TECHNOLOGY

RPC and EDA: Thrift or Tango

Remote Procedure Call (RPC) is an interprocess communication technique. It is used for client-server applications. RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction. This procedure call also manages low-level transport protocol, such as User Datagram Protocol, Transmission Control Protocol/Internet Protocol etc. It is used for carrying the message data between programs. The Full form of RPC is Remote Procedure Call.[4]

Event-driven architecture is a design model that connects distributed software systems and allows for efficient communication. EDA makes it possible to exchange information in real time or near real time. It is common in designing apps that rely on microservices. The concept of event-driven architecture is mainly realized through the publish/subscribe communication model. Publish/subscribe is a flexible messaging pattern that allows disparate system components to interact with one another asynchronously. The key point here is that pub/sub enables computers to communicate and react to data updates as they occur. This is in contrast to the traditional request/response messaging pattern where data is updated at intervals, as a response to a user-initiated request. There are always two participants — a client and a server. The client makes a call over the HTTP protocol and waits for a server to respond with the requested content.

Thrift[5] is a lightweight, language-independent software stack with an associated code generation mechanism for RPC. Thrift provides clean abstractions for data transport, data serialization, and application level processing. Thrift was originally developed by Facebook and now it is open sourced as an Apache project. Apache Thrift is a set of code-generation tools that allows developers to build RPC clients and servers by just defining the data types and service interfaces in a simple definition file. Given this file as an input, code is generated to build RPC clients and servers that communicate seamlessly across programming languages, as provided in Fig. 2.

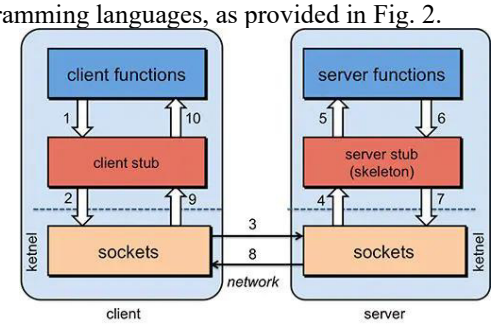


Figure 2: The software architecture of Thrift

Tango Controls is a free open source device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA (supervisory control and data acquisition) systems. Tango Controls is operating system independent and supports C++, Java and Python for all the components. Tango Controls is a hardware independent toolkit. That means you can use your driver to connect hardware with Tango Controls, as provided in Fig. 3.

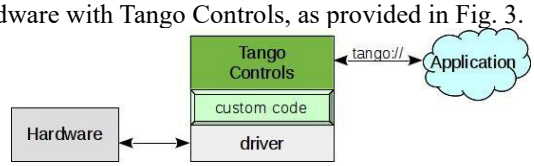


Figure 3: Tango Controls can be used as a distributed system

Script Engine: Elk[6]

Elk is a tiny embeddable JavaScript engine that implements a small but usable subset of ES6. It is designed for microcontroller development. Instead of writing firmware code in C/C++, Elk allows to develop in JavaScript. Another use case is providing customers with a secure, protected scripting environment for product customisation.

Realtime Database: LevelDB[7]

LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values. LevelDB has three basic operations: Get, Put, and Delete. Get retrieves a value given a key, Put writes a value into a key, creating the key if it doesn't exist, and Delete deletes the key and its value. There are open (takes a filename argument) and close functions for creating/loading and unloading a database, and functions that return iterators over all the keys and values. The keys and values can be any byte array and not just strings. This is useful if you have data that you want to store that you don't want to encode into a string. LevelDB supports atomic operations. You can run many operations at once in a single uninterruptible call.

COMPONENT-ORIENTED DISTRIBUTED DATA INTEGRATION MODEL

Component of Integrated Control And Data Acquisition

As provided in Fig. 4, CICADA is the Component of Integrated Control And Data Acquisition. CICADA implements a common component for data acquisition and integrated control. Data is stored in a real-time database, control logic is stored in JavaScript scripts, and the script engine is used to update data and dispatch commands regularly. It can exist as a distributed service in the local area network and be called by other component instances; it can also be used as a service access module of the client software to access other component instances; it can also be used as a server and implemented as an IO device control server; Replacing different RPC modules can be used as a bridge between different middleware software.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

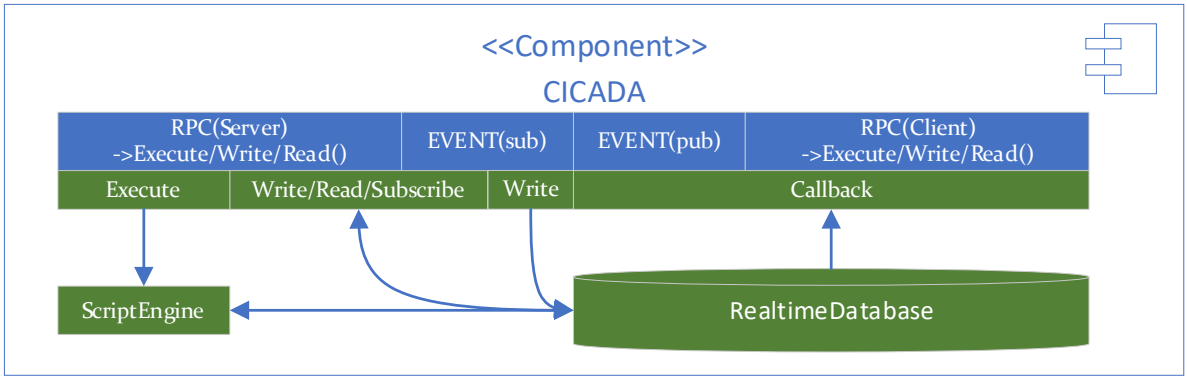


Figure 4: The composition structure of Component of Integrated Control And Data Acquisition.

The Data Centre (Realtime Database) is the data sharing centre of the components, which is responsible for storing and interacting data. The data tag can be configured with multiple attributes, what kind of behaviour is triggered after the data is executed or updated, whether the data is readable and writable, and so on.

The script engine is responsible for interpreting the executor to execute the script, interacting with the Data Centre, and realizing the logical operation of the data.

The local function set is responsible for basic functions such as executing scripts, reading and writing data, subscribing to data, and publishing data.

The RPC/EVENT module is responsible for information exchange between components, including RPC and event-driven methods, which can be used according to the application scenario, or both. The RPC/EVENT module is compatible with TANGO and Thrift software middleware, and supports expansion to CORBA or other software middleware products.

The Entity of CICADA

CICADA has three forms of existence, one is to be integrated by the client software in the form of a dynamic library, as the data acquisition and integrated control module of the client software; the other is to exist as a system service and be dispatched by the client software as a device service, and at the same time As an example of the client software scheduling other components; one way is to exist as a device service, which can be accessed by system services or client software, and part of the RPC module is replaced by the hardware driver SDK.

When it exists as a service, we design a container for CICADA. It contains an administrative module that is responsible for management functions such as creating, restarting and deleting components, similar to the DServer module in Tango software middleware.

As shown in the Fig. 5, we usually divide the control software into three layers [8]: integrated control layer, system service layer and equipment service layer. In these three-layer control software, we can all use CICADA to realize the interconnection and intercommunication part of the software at their respective levels, which greatly simplifies the many-to-many access control function between nodes without worrying about the implementation details.

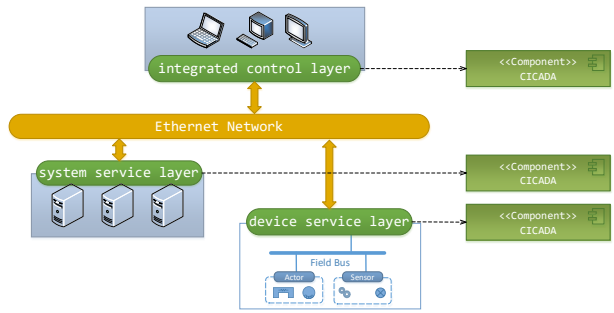


Figure 5: Use CICADA in the three-tier control software architecture.

Connection between CICADA Entities

CICADA entities exchange information and control scheduling through software middleware, as provided in Fig. 6. Components run in their own containers, which can be servers or clients. As an independent microservice or client, the CICADA entity acts as a node in a distributed network. The corresponding connection relationship is established according to the business scenario. They are loosely coupled, and the update influence domain of the component is limited, which can be guaranteed The control system software of the entire facility can evolve forward relatively stably and smoothly.

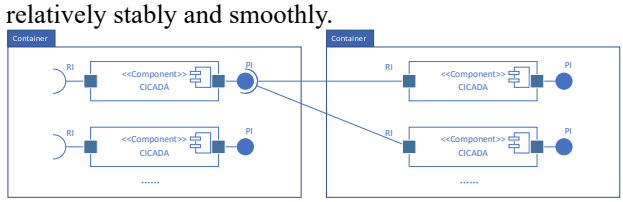


Figure 6: Exchange information and control scheduling between CICADA Entities.

Connection between early and current control systems

For example, the new system is implemented using TANGO software middleware, while the early system is implemented using Thrift software middleware. How to realize the information exchange between the early software system and the new system is a problem.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV032

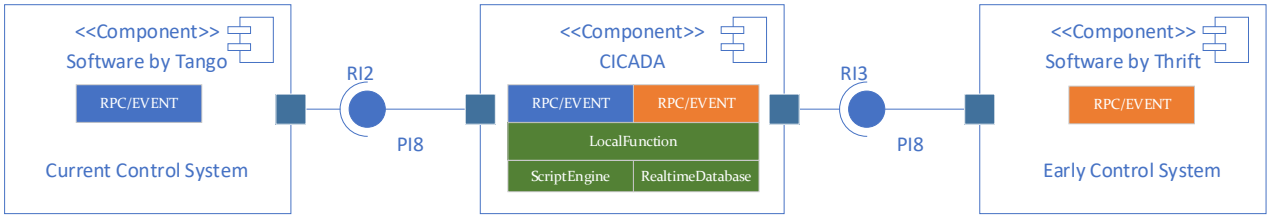


Figure 7: Use CICADA as a bridge between early and current control systems.

Now we use CICADA as a bridge between the two systems, basically zero programming can realize the information exchange function between the two software systems, as provided in Fig. 7.

CONCLUSION

Through the design and realization of CICADA components, we achieved two goals: the interconnection between the early software system and the current software system, and zero programming to adapt to the rapid changes in software requirements. Of course, we implemented this component to adapt to a workaround under specific needs. Whether it is effective for a long time still needs time to test, and this component is not considered in terms of QoS, so it needs to be strengthened in the follow-up work. Hope to communicate with colleagues, we will make CICADA components better and more practical.

REFERENCES

[1] D. J. Yao *et al.*, "Research on Software Architecture of Centralized Control System for High Power Laser Facilities, Computer Engineering and Design", vol. 28, pp. 1737-1740, 2007.

[2] EPICS Control System, http://www.aps.anl.gov/epics/

[3] TANGO Control System, http://www.tango-controls.org/

[4] RPC, https://www.guru99.com/remote-procedure-call-rpc.html

[5] Thrift, https://thrift-tutorial.readthedocs.io/en/latest/in-tro.html

[6] ELK, https://github.com/cesanta/elk

[7] LevelDB, https://github.com/google/leveldb

[8] Z. G. Ni, L Li, J. Luo, J. Liu, X. W. Zhou, "The Design of Intelligent Integrated Control Software Framework of Facili-ties for Scientific Experiments", in Proc. *17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Oct. 2019, pp. 132-136, doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV033

WEB CLIENT FOR PANIC ALARMS MANAGEMENT SYSTEM

M.Nabywaniec*, M.Gandor, P.Goryl, Ł.Żytniak S2Innovation, Cracow, Poland

Abstract

Alarms are one of the most important aspects of control systems. Each control system can face unexpected issues, which demand fast and precise resolution. As the control system starts to grow, it requires the involvement of more engineers to access the alarm's list and focus on the most important ones. Our objective was to allow users to access the alarms fast, remotely and without special software. According to current trends in the IT community, creating a web application turned out to be a perfect solution. Our application is the extension and web equivalent to the current Panic GUI application. It was developed to be integrated with EPICS and TANGO control systems. It allows constant remote access using just a web browser which is currently present on every machine including mobile phones and tablets. In that paper the status of application will be presented as well as key features.

ALARM SYSTEM IN TANGO CONTROL SYSTEM

TANGO Control System is object-oriented control system based on CORBA. It is widely used in order to create Supervisory Control and Data Acquisition system architecture. One of the most important advantages is that it is available under Open Software free license. TANGO is widely used in scientific facilities e.g. Max IV (Lund, Sweden), ALBA (Barcelona, Spain) or SOLARIS (Cracow, Poland) as well as in industry [1]. In institutes like synchrotrons the scientists and engineers are dealing with thousands of signals per second coming from different types of devices. It is clear that there is a need to monitor non-typical situations. Here comes the idea of alarm. It is asynchronous notification that some event happened, or a given state was reached. In scientific facilities using TANGO Controls the idea of creating a set of tools to manage alarms led to the creation of an alarm system. One of the most popular systems are PANIC and Tango Alarm System.

PANIC

PANIC (Package for Alarms and Notification of Incidences from Controls) was developed in ALBA Synchotron. It is a set of tools including API, Tango device and user interface for evaluation of a set of conditions and user notification. [2]

PyAlarm

One of key elements of PANIC toolkit is PyAlarm device server [3]. According to documentation, it connects to the list of alarms provided and verifies their values. Each alarm is independent in terms of formula, but all alarms within the same PyAlarm device will share a common evaluation environment. PyAlarm device allows also to configure mail or SMS notification as well as logging. That features are needed to ensure convinient alarm management for user.

Panic GUI

Panic GUI is a desktop application implemented using Taurus library. It allows checking existing alarms and manipulate them.

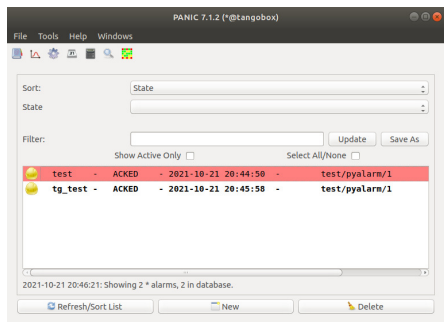


Figure 1: Alarm dashboard in PANIC GUI

Moreover, PANIC allows user to modify alarm formula, acknowledge or disable alarm.

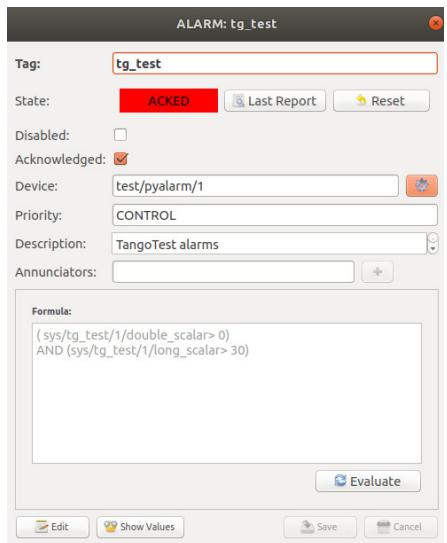


Figure 2: Alarm modification in PANIC GUI

IC@MS

Although PANIC GUI is a useful application, it has some drawbacks. First, it is desktop application what means it requires installation, and it can be only run on a desktop Nowadays, there is a trend to move desktop applications to web.which can easily by open on wider range of different devices including mobile phones, tablets and desktop

* mateusz.nabywaniec@s2innovation.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV033

Moreover, user experience is limited by Taurus library used to create interface. When creating web application, there are multiple modern frameworks which make websites look modern and easy on the eye. That are main reasons why we decided to create our web application. Our application is called IC@MS - Integrated Cloud Ready Alarm Management System. It provides the same functionalities as Panic GUI but also extends it with some new functionalities like adding devices supporting different protocols. What is important, IC@MS can be integrated not only with TANGO but also with alternative control system - EPICS [4].

KEY FEATURES

In that section we want to describe most important parts of IC@MS and explain, why they make our application so useful.

Cloud Ready

Our application is cloud ready. It means that it can be simply deployed to the cloud like Azure. It speeds up product deployment process. The application is easily accessible and is ready for future development for next versions.

Signing Up

For IC@MS security is important. Only registered users can log in and see the dashborad. Moreover, we have introduced roles to manage access to different parts of our application. Thanks to that, in production mode, people with different access levels and permissions will see only parts related to their job.

Alarm Dashboard

As well as in PANIC GUI, the most important part of application is alarm dashboard which allows seeing active and not active. When the alarm is triggered, its colour is changed what depends on severity. The most important and recent alarms are shown first. There are buttons to perform operations like acknowledging alarms, reset or disable.

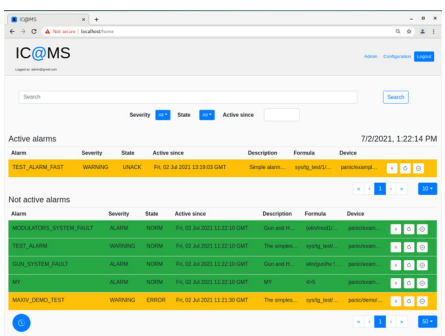


Figure 3: Alarm dashboard in IC@MS

User can easily create new alarms as well as modify existing ones using the form in the configuration page. User can include provide mail and phone number to get notifications about alarm. Moreover, IC@MS allows out of the box integration with any TANGO based device server.

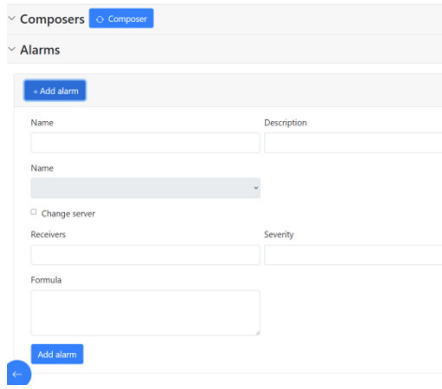


Figure 4: Creating alarm in IC@MS

Data Sources and Composers

One of IC@MS's features - the configuration page, allows to create data sources and composers devices. Data sources are TANGO devices which provide data acquired from HTTP, MQTT or Modbus protocol. For example, it can be integrated with thermometer which uses MQTT protocol, MQTT data source is created and broker IP , port, topic provided. Composer is a device which propagates signal from many low-level devices (like data sources), and can be used to define alarm formulas.

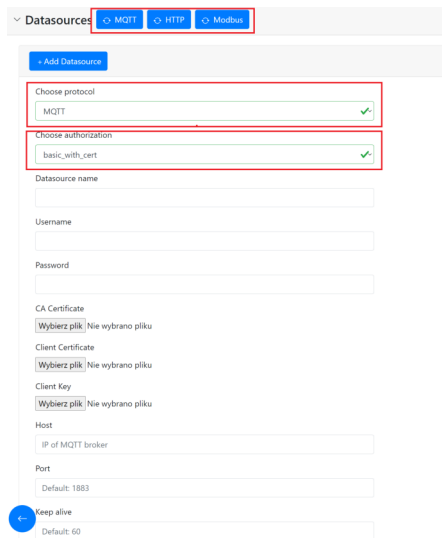


Figure 5: Creating MQTT data source in IC@MS

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ALARM HISTORY

The next feature is alarm history page, which allows user to check history of alarms including dates, state and values of formula connected to that given alarm. Moreover, user can sort entries by date or severity.

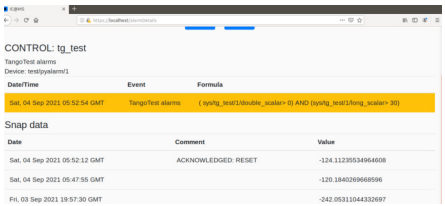


Figure 6: Alarm history in IC@MS

CONCLUSION AND FUTURE WORK

That paper shows key features of web application based on PANIC GUI and how it can be used in order to fulfil requirements of modern alarm systems. According to trend to use web applications, we see a great development potential for that project. Our plans include e.g. fixing navigation issues and extending phonebook.

REFERENCES

[1] TANGO Controls System website, https://www.tango-controls.org/.

[2] PANIC documentation, https://tango-controls.readthedocs.io/projects/panic/en/latest/.

[3] PyAlarm documentation, https://www.tango-controls.org/developers/dsc/ds/1401/

[4] EPICS website, https://epics-controls.org/.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

MIGRATION OF Tango CONTROLS SOURCE CODE REPOSITORIES

M. Liszcz, K. Kedron, P.P. Goryl, M. Celary, S2Innovation, Kraków, Poland
C. Pascual-Izarra, S.Rubio, A. Sánchez, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
B. Bertrand, MAX IV Sweden, Lund, Sweden
R. Bourtembourg, A. Götz, ESRF, Grenoble, France
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
G. Abeille, SOLEIL, Gif-sur-Yvette, France
A.F. Joubert, SARAO, Cape Town, South Africa
T. Braun, Byte Physics, Berlin, Germany

Abstract

At the turn of 2020/2021, the Tango community faced the challenge of a massive migration of all Tango software repositories from GitHub to GitLab. The motivation has been a change in the pricing model of the Travis CI provider and the shutdown of the JFrog Bintray service used for artifact hosting. GitLab has been chosen as a FOSS-friendly platform for storing both the code and build artifacts and for providing CI/CD services. The migration process faced several challenges, both technical, like redesign and rewrite of CI pipelines, and non-technical, like coordination of actions impacting multiple interdependent repositories. This paper explains the strategies adopted for migration, the outcomes, and the impact on the Tango Controls collaboration.

INTRODUCTION

Tango Controls [1] is a free and open-source software ecosystem for building distributed SCADA systems. Nowadays Tango Controls consists not only of the middleware libraries providing the core functionality but also of many supporting applications, GUI toolkits, and other utilities. Tango Controls is developed by numerous contributors and members of the Tango community.

Maintaining such a large and complex project requires best-in-class tools for planning, development, testing, packaging, and version control. From 2016 till 2021 Tango Controls was using GitHub [2] for source code management and various other services, including Travis CI for continuous integration and JFrog Bintray for publishing release artifacts. At the end of 2020, Travis CI announced the shutdown of `travis-ci.org` [3], while `travis-ci.com`, previously focused on commercial software, received a new pricing model [4]. Soon after that, JFrog announced the shutdown of Bintray service [5]. These events resulted in a mass migration of Tango Controls software repositories from GitHub, Travis CI, Bintray, and other previously used service providers. The following sections describe the migration strategy, the challenges faced during the process, the outcomes of the migration, and the impact on the whole Tango Controls collaboration.

MOTIVATION AND STRATEGIC DECISIONS

When in 2015 the tango-controls project was planning its migration out of SourceForge [6], both GitHub.com and GitLab.com were considered as its destination. In favour of GitLab was its licensing policy (GitLab maintains an Open Source Community Edition while GitHub is proprietary software), but GitHub was chosen because, it was then reasoned, its larger popularity would increase the visibility of Tango and facilitate the integration of third party services such as Travis-CI. Over the years, however, GitLab increased its user base considerably–in particular within the Tango developers because of GitLab being installed on-premises in many Tango Collaboration facilities. GitLab also greatly improved its CI integration, making it much more attractive than Travis CI (e.g. for its native support of Docker containers). Meanwhile, in 2018 Microsoft acquired GitHub –an event that already prompted many projects to move from GitHub to GitLab– and launched its own CI service called "GitHub Actions" [7] which also overcame many limitations of Travis CI. The definitive trigger for considering the migration of the tango-controls organization came at the end of 2020 with the announcement that Travis CI would start charging for CI time also to FOSS projects [4], which directly affected the key repositories in tango-controls.

GitHub Actions or GitLab CI?

At this point, two alternatives were considered: either move the CI of the Tango projects to GitHub Actions or move the whole project to GitLab (other possibilities such as using GitLab for the CI but keeping the project infrastructure in GitHub were also considered but quickly discarded as impractical).

Obviously, staying in GitHub and only adapting the CI would entail less effort and, from the technical point of view, the GitHub Actions service was considered to be on-par with GitLab CI in terms of integration and features. However, the proprietary nature of GitHub weighted against investing effort on further integrating with it because of concerns about potential vendor lock-in. In contrast, the fact that GitLab's Community Edition is Open Source and that the Tango Collaboration members are already experienced in both using and maintaining their own GitLab instances, eased the concerns even about the eventuality that GitLab.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

could restrict its current allowance of free CI time and other support for FOSS project [8].

When evaluating the extra effort required for migrating to GitLab *vs* staying in GitHub, the project import feature provided by GitLab was found to do an excellent job in automatically transferring a whole GitHub project (including its code, issues, wikis, etc). Therefore, it was concluded that the main effort for the migration of a project lies in adapting its CI, which would had to be done also if moving to GitHub Actions, since continuing on Travis CI was not an option.

Regarding the disruption due to the change for the individual users, it was considered to be heavily mitigated by the fact that GitLab's UI is very similar to that of Github, that most of our users were already using GitLab instances on-premises and that GitLab.com allows to log-in using existing GitHub credentials.

Finally, it was also considered that by using the same platform for the collaboration as that used locally in the member facilities would be beneficial for reusing code, configurations and skills, and it would facilitate transferring projects to and from the local GitLab instances and sharing knowledge within the collaboration.

MIGRATION PROCESS

The various projects hosted in the tango-controls GitHub organization involve different subsets of the Tango Community, with different release cycles and development constraints. For this reason each project was allowed to choose its own schedule for the actual migration, but a "migration team" was constituted to coordinate the migrations, support the project administrators and provide both a set of tools and a common procedure for migrating.

The generic procedure for migrating a given project was implemented as follows:

1. If the project uses Continuous integration, its Travis and/or GitHub Actions .yml files are translated to an equivalent gitlab-ci.yml file and the result is tested on a temporary GitLab fork of the repository.

2. A list of participants to the project is compiled by means of a custom-made python script [9] that uses the GitHub REST API to scrape the account names of those who contributed code or participated in the discussions associated to Pull Requests and Issues of this project.

3. A migration date is set by the project administrators and the migration team. Then the participants identified in the previous step are notified that, at least 24h before the migration date, they should ensure that their GitHub account can be associated to a GitLab account (in practice this requires either that they login once into GitLab using their GitHub credentials or that they ensure that their GitHub account is set with a public email address also used in their GitLab account).

4. On the decided date, the migration team performs the actual migration as follows:

   (a) A new empty branch called "*moved-to-gitlab*" is created in the GitHub repository and set as its default branch. A single README file is added to it with a link to the new location in GitLab. The project description is also changed to "Moved to gitlab".

   (b) The GitHub project is "archived" (i.e., set as read-only)

   (c) The GitHub project is imported as a subproject in the GitLab.com tango-controls namespace. This is done using a dedicated "*tango-controls-bot*" account and the standard import tool provided by GitLab, which imports the whole git repository, the issues, the pull requests (merge requests in GitLab), the milestones, the releases the wiki, etc.

   (d) The *moved-to-gitlab* branch is removed from the fresh new GitLab repository and the proper default branch (e.g., "*main*" or "*master*" or "*develop*") is set instead.

   (e) The Tango Community is notified about the project migration via an email and a comment in a dedicated issue.

5. After the migration, the project admins may need to check and adjust the settings of the migrated project in GitLab (e.g., the project members, protected branches, etc.) and also manually adapt the documentation by replacing references to GitHub.com by the equivalent ones in GitLab.com.

C++ Core Library and Related Projects

The migration of the C++ core library repository [10] to GitLab presented multiple challenges due to many interactions with various external services. Travis CI [11] was used for builds on Linux platforms and regression testing, Appveyor [12] was used for builds on Microsoft Windows, Coveralls [13] provided code coverage reports, and Sonar Cloud [14] calculated code quality metrics. Additionally, the ABI compatibility check was implemented using GitHub Actions.

Linux CI The jobs on Travis CI were building and testing the Tango Controls C++ core library on multiple versions of the Debian operating system inside Docker containers. The use of containers was dictated by the limited choice of operating systems available in Travis CI [15]. The images of these containers were maintained by the Tango Controls C++ core library developers and included all the software required to build and test the library. Whilst GitLab Runner provides a Docker Executor and is already running all test jobs inside containers, it was decided to keep existing Debian-based containers and run them using

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

Docker-in-Docker [16]. The reuse of the containers allowed to reduce the effort required for the migration. The adoption of Docker-in-Docker increased the complexity of the job configuration but it could not be avoided as currently Docker is anyway required to run the C++ core library tests.

Limited hardware resources of GitLab Build Cloud runners imposed an additional challenge on the migration process. The runners offer just one CPU core [17], compared to two cores available in Travis CI virtual environment [18]. This limitation resulted in failures of some timing-dependent tests designed for multi-core systems. Such tests required case-by-case analysis and minor changes in the implementation.

Windows CI Windows-based jobs from Appveyor were not migrated to GitLab CI/CD. Instead, Appveyor has been configured as an external CI provider for GitLab. There are plans to migrate these jobs to GitLab's Windows Shared Runners, but more work is still needed to enable this migration [19].

External Services During migration, it was decided to stop using external services for code coverage reporting and code quality metrics. For both use cases, GitLab already provides built-in features that integrate better with the rest of the platform. Coveralls was replaced with a CI job that produces a code coverage report in a standard Cobertura XML format. This report is then processed by GitLab and used for test coverage visualization in GitLab's merge request UI [20]. Sonar Cloud was replaced with a CI job that runs clang-tidy [21] and produces a Code Quality report artifact [22]. This allows GitLab to show any code quality changes in the merge request UI.

Related Projects The C++ core library developers also maintain a set of device servers and utilities implemented in C++ like DataBase [23], Starter [24], TangoTest [25], TangoAccessControl [26], tango-idl [27], or tango_admin [28]. These projects do not use continuous integration or any external services, thus their migration followed the standard process of setting up appropriate "moved to" branches and importing the projects in GitLab's web UI.

The Tango Source Distribution [29] is another project closely related to the C++ core library. It used a Travis CI job to build an install-able package with the C++ library, essential device servers, and a set of Java-based tools. For the purpose of the migration, a new CI job was created in GitLab. As Ant tool [30] was used to drive the packaging process, translating the job definition from Travis CI to GitLab CI/CD did not require many changes. Additionally, the resulting package is now stored as a job artifact.

Python Core Library

The Python core library [31] was using Travis CI service for compilation and tests with multiple versions of Python, including 2.x and 3.x releases. All test jobs were utilizing Conda environments [32] to pull the required dependencies and set up a matching version of Python. Since Conda is also available as a Docker image suitable for use in GitLab CI/CD the migration process was relatively straightforward and required only translating the CI pipeline configuration into a format accepted by GitLab. The project was not using any other external services, which simplified the migration. The packaging process was not impacted and the PyTango package is still published to PyPi [33].

Java Core Library and Related Projects

Java core libraries, including JTango, use Apache Maven [34] as a software project management tool to test, build and validate the package. Maven is also used to produce binary artifacts like JAR files. Testing and building the software was automated using Travis CI service that validated all the changes to the source code which was stored in GitHub. The main challenge during migration to GitLab was the rewrite of the CI pipeline into the format expected by Gitlab CI/CD.

During the migration process, the JTango CI/CD pipeline was completely redesigned to optimize the execution time and to automatically deploy artifacts into a remote repository. Also, the obsolete and duplicated CI jobs were removed. With Travis CI, artifacts were deployed automatically to Bintray [35] and manually into Maven Central [36] repositories, as they are very popular in the Java/Maven community. Because of the Bintray shutdown [5], a new location was needed to store packages and artifacts. The free Sonatype Nexus OSS repository hosting [37] was selected for this purpose as it is a popular way to release libraries to Maven Central. As a result, both snapshot and release versions of the Tango Controls core Java libraries are now automatically deployed to OSS and synchronized with Maven Central.

Documentation

Tango Controls documentation is written in the Sphinx-compliant [38] reStructuredText format [39]. The documentation is published on the ReadTheDocs.io service [40].

The sources were kept on GitHub along with other Tango Controls projects. The project used Travis CI to test the documentation build process and to assure its quality in terms of warning-free builds, among others.

Moving the sources to GitLab was straightforward with the use of prepared scripts. In addition to setting up GitLab CI/CD (providing gitlab-ci.yml configuration file), Sphinx configuration file (conf.py) has been updated. The ReadTheDocs.io project was also reconfigured to reflect the new repository location.

The new CI pipeline takes advantage of the features provided by GitLab CI. For each merge request the documentation, including proposed changes, is built and the resulting HTML files are stored as CI job artifacts [41]. This gives the reviewer ability to check the visual quality of

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the merge request directly in GitLab without doing a local build of the documentation on the reviewer's machine.

Other Projects

Other software projects related to the core have been already migrated to GiLab or are in the process of migration (Taurus [42], Sardana [43], fandango [44], panic [45], HDB++ [46]). A developer has been assigned to the migration task of each of these projects. Some of the already migrated projects (fandango, panic) are also in transition from Python 2 to Python 3. The work done on GitLab migration allowed to implement CI/CD on these projects, taking advantage of the Docker images and jobs already configured. Migration of HDB++ Archiving project and the development of Docker images for archiving CI are assigned and expected to be completed in the next year.

IMPACT ON PACKAGING

Debian

Using GitLab enables a direct reuse of the CI pipelines from the official Debian packaging infrastructure [47] which is also based on Gitlab-CI.

RPM

The Tango RPMs used to be built internally by MAX IV. With the migration to GitLab, we took the opportunity to move the Tango SPEC file repository [48] to the tango-controls group on GitLab. RPMs are now built using Copr [49], a build system and infrastructure provided by Fedora. This could have been done on GitHub as well, but using a gitlab-ci pipeline to trigger and test the Copr build made the task easier, as MAX IV has a lot of experience with their internal GitLab server. RPMs are built using the Tango Source Distribution release [29]. This is a single url to change in the SPEC file when next release will be available.

Conda

Conda [32] packages are built using an archive of the repository. The repository url is specified in the *source* field as well as in the *metadata* of the recipe. The impact of the migration to GitLab is thus minimal. Some packages, like tango-database, were created after the migration and already use GitLab as source. Some, like cpptango, were created before. They will be updated when a new version is released.

Bintray

Java core libraries were using Bintray [35] to store binaries and other artifacts like POM files. Publishing to Bintray was the main way of distributing JTango to downstream projects. After Bintray shutdown [5] and repositories migration a GitLab package registry was created for the JTango project [50]. This registry allows for storing JAR files for each JTango release. Additionally, integration of GitLab CI/CD with Maven allowed to refactor the build process and to distribute the binaries directly to OSS Sonatype Nexus [37] which is currently the official remote registry for distributing the JTango Java artifacts.

CONCLUSION

The migration process presented maintainers and developers with many challenges. A common, well-defined strategy and a detailed migration plan allowed for carrying out all necessary activities on time and consistently across multiple projects. A dedicated migration team provided support for administrators of various repositories and coordinated all the efforts related to the migration. Move from Travis CI to GitLab CI/CD caused some technical difficulties but in the end, it allowed for better integration with the rest of the GitLab platform.

The migration of Tango Controls software repositories to GitLab is still a work in progress. At the time of writing 49 out of 67 repositories in the tango-controls organization were already migrated. Critical projects like core software libraries are on GitLab since early 2021 and all development including issue handling, code review and continuous integration is performed there. Remaining projects will be gradually migrated over time or left on GitHub if there is no clear benefit in moving them.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Tango Controls Collaboration for funding a number of the developments described here as well as the Tango Controls Community for bug reports, fixes, suggestions for new features and contributions.

REFERENCES

[1] https://www.tango-controls.org

[2] https://github.com/tango-controls

[3] https://mailchi.mp/3d439eeb1098/travis-ciorg-is-moving-to-travis-cicom

[4] https://blog.travis-ci.com/2020-11-02-travis-ci-new-billing

[5] https://jfrog.com/blog/into-the-sunset-bintray

[6] https://sourceforge.net/projects/tango-cs/

[7] https://github.com/features/actions

[8] https://about.gitlab.com/solutions/open-source/

[9] https://github.com/tango-controls/gitlab-migration-tools

[10] https://gitlab.com/tango-controls/cppTango

[11] https://travis-ci.org/github/tango-controls/cppTango

[12] https://travis-ci.org/github/tango-controls/cppTango

[13] https://coveralls.io/github/tango-controls/cppTango

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

[14] https://sonarcloud.io/dashboard?id=org.tango-controls:cpp-tango:tango-9-lts

[15] https://docs.travis-ci.com/user/reference/overview/#which-one-do-i-use

[16] https://hub.docker.com/_/docker

[17] https://docs.gitlab.com/ee/ci/runners/build_cloud/linux_build_cloud.html

[18] https://docs.travis-ci.com/user/reference/overview/#virtualisation-environment-vs-operating-system

[19] https://gitlab.com/tango-controls/cppTango/-/issues/731

[20] https://docs.gitlab.com/ee/user/project/merge_requests/test_coverage_visualization.html

[21] https://clang.llvm.org/extra/clang-tidy/

[22] https://docs.gitlab.com/ee/user/project/merge_requests/code_quality.html#implementing-a-custom-tool

[23] https://gitlab.com/tango-controls/TangoDatabase

[24] https://gitlab.com/tango-controls/starter

[25] https://gitlab.com/tango-controls/TangoTest

[26] https://gitlab.com/tango-controls/TangoAccessControl

[27] https://gitlab.com/tango-controls/tango-idl

[28] https://gitlab.com/tango-controls/tango_admin

[29] https://gitlab.com/tango-controls/TangoSourceDistribution

[30] https://ant.apache.org/

[31] https://gitlab.com/tango-controls/pytango

[32] https://docs.conda.io

[33] https://pypi.org/project/pytango/

[34] https://maven.apache.org/

[35] https://bintray.com/

[36] https://search.maven.org/

[37] https://oss.sonatype.org/

[38] https://www.sphinx-doc.org/

[39] https://gitlab.com/tango-controls/tango-doc

[40] https://tango-controls.readthedocs.io/en/latest/

[41] https://docs.gitlab.com/ee/ci/pipelines/job_artifacts.html

[42] https://gitlab.com/taurus-org/taurus

[43] https://github.com/sardana-org/sardana

[44] https://gitlab.com/tango-controls/fandango

[45] https://github.com/tango-controls/PANIC

[46] https://github.com/tango-controls-hdbpp

[47] https://salsa.debian.org/salsa-ci-team/pipeline

[48] https://gitlab.com/tango-controls/tango-spec

[49] https://copr.fedorainfracloud.org/coprs/g/tango-controls/tango/

[50] https://gitlab.com/tango-controls/JTango/-/packages/1959228

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV035

DEVELOPMENT OF ALARM AND MONITORING SYSTEM USING SMARTPHONE

W.S Cho*, Pohang Accelerator Laboratory, Pohang, Republic of Korea

Abstract

In order to find out the problem of the device remotely, we aimed to develop a new alarm system. The main functions of the alarm system are real-time monitoring of EPICS PV data, data storage, and data storage when an alarm occurs. In addition, an alarm is transmitted in real time through an app on the smartphone to communicate the situation to machine engineers of PLS-II. This system uses InfluxDB to store data. In addition, a new program written in Java language was developed so that data acquisition, analysis, and beam dump conditions can be known. furthermore Vue.js is used to develop together with node.js and web-based android and iOS-based smart phone applications, and user interface is serviced. Eventually, using this system, we were able to check the cause analysis and data in real time when an alarm occurs. In this paper, we introduce the design of an alarm system and the transmission of alarms to an application.

INTRODUCTION

The off-line method of the past alarm system sets threshold points for Low, High, and Interlock information, outputs a sound through the speaker when the range is out of range, and the operator immediately analyses the contents and solves the cause. This is a useful solution in field situations. However, engineers cannot see real-time data or alarm information generated by the device.

The new real-time online alarm method selects the EPICS PV [1] data requested by engineers or users in addition to the specified threshold and checks the data or receives an alarm with their smartphone. The new alarm system uses multiple user subscriptions based on EPICS PV. If the user wants to receive an alarm or there is data that he wants to monitor in real time, the PV is registered in the alarm system through the smartphone application. For example, the beam current data of the storage ring is unique. However, individual users may have different desired alarm setting conditions. That is, some users may want to be notified when the current is less than 100mA, and another user may want to receive an alarm when the current is less than 50mA. The alarm system is completely individualized for each user so that alarms are sent or stopped according to different conditions required for each as shown in Figure 1. And when an unexpected failure occurs, the PV connection of the EPICS IOC may be disconnected, so the EPICS PV connection information can also be included in the alarm. The smartphone application to receive the alarm can be used in both Android and iOS environments. We chose Web App because it is important to reduce development time and simplify maintenance. Among the many frameworks used in Web App, Vue.js [2] is applied. HTML5 and Java script can be easily handled, templates are modular, and switching to mobile or desktop environments is simple. Data can be checked up to 5 hours ago from the present using a line chart. In addition, to check the data of EPICS PV, the data value displayed in the application is updated in real time once every 3 seconds using Web Socket.

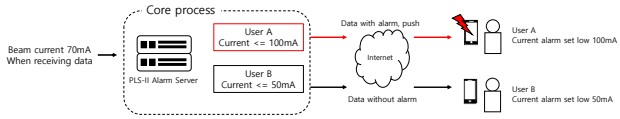


Figure 1 : Alarm system operation overview.

ALARM SYSTEM

Each embedded device provides the user with the necessary data. These data are integrated through the EPICS Gateway server. The alarm system connects to the gateway server through a network by a process called the integrated alarm system core and collects data at a rate of up to 10Hz according to the data cycle updated for each device.

The EPICS JCA library was used to collect PV connection status and data by connecting to the IOC through the network. Users who use the alarm system select a PV to receive real-time monitoring and alarm services through a smartphone application. And when parameters linked to PV are set and saved, data monitoring starts immediately in the core process. All data is stored in InfluxDB while monitoring data. And the data is processed by analysing the parameters set by each user, and if the alarm condition is met, an alarm is immediately prepared to be sent to the user. Alarm information is stored in an event queue running as an asynchronous thread inside the core process. When data exists in the event queue, it immediately analyses the event contents and stores the event log and data from the last hour or so in SQL and the file system. When all processing is completed, a PUSH notification service is sent to an Android or iOS smartphone through Google Firebase service. When the user receives the PUSH notification service, the notification of the installed application is generated, and when the user selects the corresponding event, analysis can be performed together with the past data.

Integrated Alarm System Core

Integrated alarm system core is developed in Java language and executed as a process inside O/S. This process is composed of 5 functions, and the role is divided for each function as shown in Table 1.

* wscho@postech.ac.kr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV035

Table 1: Integrated Alarm System Core Function

Function	Description
PV Status	Check PV status
Data Watcher	Data update interval monitoring
Event Queue	Alarm event information storage
Firebase	Message push service
Web Socket	Real-time data update support

PV status provides the user with IOC PV communication connection status information and PV data. If the PV communication connection state is disconnected, the IOC or internal network situation is not good. In this case, a notification is first sent to the user so that user can immediately check the IOC status or network status.

PV data watcher measures the time of data acquired at regular intervals. This is necessary because IOC works fine, but I have had experience with data not being updated. IOC acquires data through serial or ethernet connection with the device and transmits the data to PV. The general IOC data update period can be set through the SCAN of the EPICS record field. However, if the data is not updated for the time interval set by the user separately from the EPICS field information, it can be inferred that there is a problem in the connection between the IOC and the device. It is not easy to detect this communication problem so far, but through the newly developed alarm system, if data is not updated for a set period of time, it is possible to quickly identify any problem in the connection.

The event queue stores the user ID, time, value, and PV name in the database when the alarm condition is satisfied based on the collected data as shown in Figure 2. And it can be seen that the reason for the need for a user ID is that alarm sending conditions may be different for each user based on the same data.

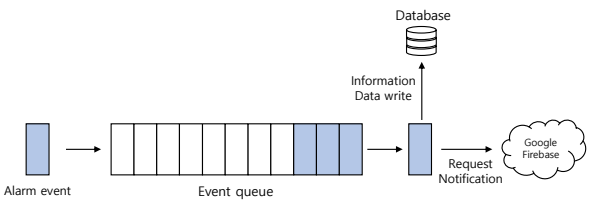


Figure 2: Event queue in alarm process.

Google Firebase [3] provides an integrated environment to send notifications including event contents to various client environments such as Android, iOS, and web browsers.

WebSocket is a technology for the purpose of communication between the user's web browser and the server. It is used in the alarm system of PLS-II and supports sending and receiving messages freely in both directions. Therefore, it supports the function to monitor the current PV data in real time to the user. When the user runs the smartphone application, the user can check the PV name registered in advance and the value updated in real time, so that the user can check the situation in real time.

The inside of the class of the core process consists of PV name, user information, value, time information, and various functions. When the user sets the PV to be monitored, the requested PV creates a new class inside the core process. If another user requests a previously used PV, user information is added from the existing class. When the class is created, PV connection is immediately proceeded, and the following pseudo code is executed according to the alarm range specified by the user.

Pseudo code 1: Alarm condition

```
Data stored in influxDB
if (data < low or data >= high) AND flag then
   if nowTime – lastTime > intervalTime then
      alarm push
   end if
end if
```

The pseudo code monitors the PV and when the value is updated, it is saved in influxDB. And according to the first conditional statement, if the range of values is within the range set by the user and it is true to receive an alarm, the next conditional statement is executed. After subtracting the last stored time from the current time, it is compared with the alarm update time set by the user to determine whether to send an alarm. This function uses a simple algorithm in order not to send a large amount of alarm to the user's application whenever the value satisfying the alarm condition changes. For example, the current time is 10 and the previously stored time is 5. And it is assumed that the user sets the interval time to 3. In this case, since the condition is true, an alarm is sent to the user. If the user's interval time setting value is 10, the range of the value satisfies the alarm condition, but it is not transmitted to the user.

Back-end (Server side)

Back-end performs a web server function that responds by processing user requests based on Node.js framework. Web server functions are divided into two main categories.

The user's smartphone application request is processed using RESTful API that is processed in HTTP protocol environment and performs commands such as POST and GET. Node.js can develop programs by distinguishing POST and GET commands from the same route address. When a user requests data, the GET function is used.

The user register function creates a token used by Google Firebase in the smartphone application when a new user is created. The token value is needed to send a notification to the user in the alarm system core process. When the token is sent to the web server through a specified route, the server issues a unique user ID that is a combination of alphanumeric characters and sends it to the application. Based on the issued user ID, the web server processes the inquiry request and provides data to the user.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV035

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The alarm history function can be checked by inquiring the alarm information previously stored by the alarm system core process. Alarm information is information stored in the file system based on the user ID and unique time of the PV registered by the user up to 5 hours ago. When the user selects the time information, the stored data is immediately loaded by the web server and the data is transmitted to the user application. Past data shows the entire PV registered by the user. That is, it is assumed that all PVs registered by the user are related.

Create and delete PV monitor registers or deletes the PV to be monitored by the user. When user registers PV, alias, formula application, LOW, HIGH, alarm interval, Y-axis log scale, alarm reception, etc. can be set. And the user PV update function can modify the values set during registration.

User PV list receives the list of all PVs registered by the user. If you use this function, you can check the real-time data along with the PV name in conjunction with the WebSocket session.

It maintains security between the server and the application. When providing web services, external attacks occur, so security must be set. Two-way communication data must be encrypted by setting SSL (Secure Socket Layer) and WSS (WebSocket Secure) as security settings. And it protects from external threats by altering HTTP server information and showing other server information to the attacker.

Front-end (Client side)

Vue.js is a front-end framework and supports SPA (Single Page Application), so it can play the role of a smartphone application. And since it is possible to access the DOM (Document Object Model) interface in an easy way, it is possible to selectively update only the parts that need to be changed in interaction with the user. In addition, since it uses a modular method, it can be easily applied to other templates if a specific page or function is developed.

As a security function of Vue.js, the history function displays an error screen when an arbitrary route is connected without going through a set route.

In the layout design of the smartphone application, as shown in Figure 3, three bottom buttons are applied: HOME, HISTORY, and SETTING.

HOME shows a list of all PVs registered by the user and values that change in real time. If user select a PV name or an alias, user can check the data up to 5 hours ago in a graph. Therefore, even if a specific alarm situation does not occur, users can easily check the real-time information they want anytime, anywhere.

HISTORY shows the past records of alarms that have occurred to the user by alarm occurrence time, value, and PV name. If the user selects this list, user can check the historical data stored before the alarm in a graph.

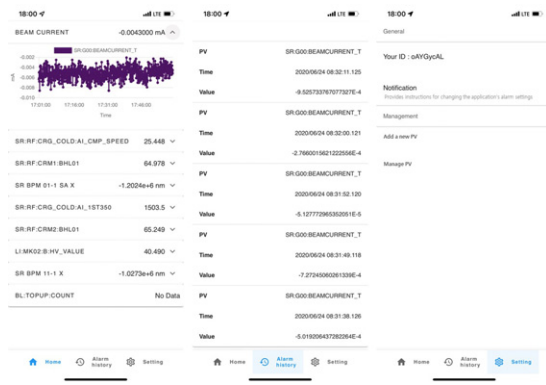


Figure 3: Smart phone application view (iOS).

SETTING can change the user's application usage settings. When an alarm occurs to the user, it is possible to turn on/off the function that the alarm information is exposed at the top of the smartphone screen through the notification function of the application. In addition, the function to add, edit, or delete PVs to be monitored can be performed.

CONCLUSION

In this paper, we introduced a system that supports the real-time alarm function based on a smartphone application. It has become possible to support monitoring functions by individualizing the data of devices that users want and to respond quickly to failures. The read/write performance of InfluxDB time series data in the PLS-II alarm system shows satisfactory performance, but the clustering system is replaced and We plan to upgrade the performance of the data archiving system of PLS-II to collect more data through integration with the alarm system. Currently, the notification function is supported by using a simple algorithm that compares the specified threshold, but it aims to further improve the system through ML (Machine Learning) in the future. The main goal of ML is to predict and detect abnormal operation of the device in advance to strengthen the protection function. When the abnormality detection technology reaches the required level, it will be possible to develop a better system than now by interworking with the alarm system.

ACKNOWLEDGEMENTS

I would like to thank the members of the PLS-II Accelerator divisions for their comments on the system and for testing the program.

REFERENCES

[1] EPICS, https://epics.anl.gov/

[2] Vue.js, https://vuejs.org

[3] Google Firebase, https://firebase.google.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV036

PORTING CONTROL SYSTEM SOFTWARE FROM PYTHON 2 TO 3 - CHALLENGES AND LESSONS

A. F. Joubert, M.T. Ockards, S. Wai, SARAO, Cape Town, South Africa

Abstract

Obsolescence is one of the challenges facing all long-term projects. It not only affects hardware platforms, but also software. Python 2.x reached official End Of Life status on 1 January 2020. In this paper we review our efforts to port to the replacement, Python 3.x. While the two versions are very similar, there are important differences which can lead to incompatibility issues or undesirable changes in behaviour. We discuss our motivation and strategy for porting our code base of approximately 200k source lines of code over 20 Python packages. This includes aspects such as internal and external dependencies, legacy and proprietary software that cannot be easily ported, testing and verification, and why we selected a phased approach rather than "big bang". We also report on the challenges and lessons learnt - notably why good test coverage is so important for software maintenance. Our application is the 64-antenna MeerKAT radio telescope in South Africa — a precursor to the Square Kilometre Array.

INTRODUCTION

The MeerKAT radio telescope [1] is operational in the Karoo region of South Africa, with its 64 dish antennas. It is a precursor to the larger Square Kilometre Array project [2], and will be integrated into the mid-frequency array, SKA1 MID.

The focus of this work is to report back on our approach, and progress related to the effort of porting our codebase from Python 2 to Python 3 [3]. The telescope has many subsystems with their own teams and software codebases — here we look at the control and monitoring software system.

This paper is organised as follows. First, we briefly compare Python 2 and Python 3. We then discuss the motivation for change and strategies considered. The actual approach taken, and the results are then presented, before concluding.

PYTHON 2 VS. 3

The main driver leading to the creation of Python 3 was to clean up problems with the language [4]. These changes were backwards incompatible by design. The most fundamental change is the representation of strings using Unicode by default, rather than 8-bit ASCII. This resulted in a clear split between binary data and text data.

There are a number of other changes [5], including: `print` became a function, module imports are now absolute, rearrangement/renaming of the standard libraries, more use of generators instead of concrete containers, division automatically promotes to floating point, classical classes are removed, and first-class support for asynchronous coroutines with `async` and `await`.

MOTIVATION FOR CHANGE

Porting a codebase requires significant effort, and any changes introduces risks like bugs, downtime, and degraded performance. The benefits of porting need to outweigh these negative factors.

Positive factors:

- Python 2 is end-of-life since January 2020. This means no further changes or fixes to the interpreter will be provided.
- The ecosystem of Python packages are dropping Python 2 support. This also means no fixes, no security updates, an inability to benefit from future improvements in existing packages, and new packages. Another problem is that dependencies with unpinned requirements start to break, when a new version of a package adds Python 3-only features incorrectly.
- Linux distributions are no longer including packages for Python 2 [6]. While there are ways of installing Python 2 now, this is likely to get more and more difficult. A lack of Python 2 support on future Linux releases, means that we cannot upgrade our servers, and are "stuck" on old operating systems.
- In our opinion, software engineers tend to favour newer tools, and Python 2 is now considered "legacy". There are new features in Python 3 that improve the development experience. Software engineers that enjoy the work they do are more motivated and committed to their projects.
- The remaining lifespan of MeerKAT, before it is consumed by SKA, is at least 5 years, so we need to be able to continue improving the software for at least that long.
- Revisiting the whole codebase while porting helps to increase the knowledge and understanding of the codebase, and can result in old bugs being discovered and fixed.

Negative factors:

- A huge amount of work is required, so other software features/improvements have be delayed in order to allocate human resources to the porting project.
- Risk of introducing bugs which lead to incorrect operation, downtime or degraded performance.

Overall, we believed that the positive aspects make the exercise worth doing.

STRATEGIES FOR PORTING

The Python porting book [7] outlines a few strategies for porting to Python 3.

Unfortunately, most of the projects in our codebase are libraries and therefore coupled during runtime (they must run

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV036

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

in the same Python environment with shared site-packages). This precludes us from any "big bang" approaches moving directly to python3, so our upgrade path will have to be staggered, by porting one package at a time.

The remaining approaches will be discussed below, ending with our selection.

Separate Branches for Python 2 and Python 3

Maintaining two different branches and distributing two separate versions of the code. This is where the most effort and complexity involved. This would entail patches (pull requests) to the python2 codebase and a simultaneous patch/PR to the python3 with equivalent changes. In addition, the python2 codebase also has to be ported to python3 initially.

This is the most complex and expensive option.

Single Source (Python2), Two Distributions (Python2 and Python3)

Keep the source in python2 and use python-modernize [8] to automatically convert the source to python3 compatible distribution packages. The python-modernize package wraps a tool called 2to3 [9]. There will be some upfront effort required to write "fixers" for code that cannot be automatically converted and some scripts to ensure conversion happens for every new change. Eventually, the handwritten codebase will have to be replaced by the generated python3 code.

Where's the complexity? Verifying correctness of automated conversion and writing fixers or manually editing generated python3 code for edge cases. Some learning curve around 2to3 and distutils.

What's the benefit? Since this is an additive change, the current way of developing can remain the same on python2. This allows the team to ease into python3.

Risks The python3 code remains a second class citizen and becomes maintained/de-prioritised. Code that is never run eventually becomes dead code.

Single Source, Single Distribution (Python2 and Python3 Wrapper)

Using a wrapper layer and maintaining a codebase that is compatible in both python2 and python3. This entails leveraging linting and type-checking tools to ensure new changes are compatible with both and sometimes writing code that delegates to a wrapper/compatibility layer such as six [10]. We'd still want to use python-modernize or python-future [11] to do the initial conversion of the code to a compatible state.

Where's the complexity? The compatibility layer might create a "type explosion" - e.g., python-future includes additional string and byte types. The toolchain would have to be very strict and robust to detect and prevent any incompatible changes (linting, type-checking by pylint, mypy, pytype, etc.). The initial conversion *might* be trickier and more complex than simply writing python3 code.

What's the benefit? A single codebase, albeit written in a strict style with a intersection set of python2 and python3 features.

Risks Some additional complexity in write-time and potentially ugly wrappers. A possibility that the Python 2-compatible code lives on forever.

Single Source (Python3), Two Distributions (Python2)

Maintain the source in Python3 and convert to be Python2 compatible via automation. There will be upfront effort to convert the projects to strictly Python3 but (it also means a clean break into python3 when the time comes).

Where's the complexity? We'd have to convert (and verify) an entire project upfront to python3. 3to2 conversion fixers would have to be written and maintained, however it's most likely easier to introduce python2 compatibility to python3 code than vice-versa and conversion is throw-away effort.

What's the benefit? Once a package is upgraded to python3, we can simply turn off the build to python2 when the time comes. It's also most likely easier to introduce python2 compatibility to python3 code than vice-versa.

Risks Upfront effort to convert some packages might be huge.

Strategy Selection

The two-branch strategy was considered too time consuming to pursue - all patches (pull requests) would have to be duplicated for each branch. That left 3 strategies, with the big difference between being whether or not to autogenerate production code. We decided against autogeneration, as this would result in two codebases (original, and autogenerated) that need to be verified by humans. Our code coverage from unit tests was not considered sufficient as the sole method of verification for the autogenerated code.

Thus the porting strategy selected was *Single source, single distribution (python2 and python3 wrapper)*. For the wrapper library, we chose python-future instead of six. They are quite similar, so there was not a strong deciding factor here.

APPROACH

The details of the approach followed are described in this section. We look at the package dependencies, the size of the codebase (lines), and finally discuss the workflow adopted.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV036

Dependencies

There are several Python tools that can be used to determine a dependency graph for packages, including: pydeps [12], snakefood [13], and pipdeptree [14]

pipdeptree was chosen because it leverages pip [15] to determine dependencies on a package level, rather than file level (unlike snakefood which operates on file level). It also supports a `-reverse` flag (unlike pydeps) which shows which downstream packages require a given dependency, i.e. why that particular package is installed. This means we can limit our analysis to our own codebase's packages.

The goal was to rank packages based on their place in the tree. Packages that are dependencies for many others downstream are more risky whereas packages that are leaf nodes are less risky. However, our porting needs to start at the root of the dependencies (i.e., most used package). If we try to port a package with a dependency that only supports python2, we will not be able to test that package under python3, so we will have no confidence in the port.

Leaf nodes are typically where our application scripts are found. They make use of various libraries higher in the dependency tree to build the required functionality. Only when a leaf node has been ported can we start running some of the applications that make up our distributed control system under Python 3.

Unfortunately, the default pipdeptree output is in the form of text. There is a flag to output graphviz [16] `.dot` files but it does not support the `-only` flag to filter out packages we are interested in.

Some modifications are required to pipdeptree to make it work for our purposes. So pipdeptree was patched [17] accordingly. The resultant dependency graph is shown in Fig. 1. Our approach was to work from top to bottom, to ensure that we have python3-compatible versions of all dependencies.

Lines of Code

As part of estimating the complexity of the work, we analysed code metrics, including the number of lines of code. This was done using a Radon [18]. The metric of interest was the number of source line of code (SLOC).

The total for all packages was 214 k SLOC, with the smallest package around 0.15 k SLOC, and the largest around 41 k SLOC. The average size was 9.7 k.

We intended to use these numbers to help with estimating the time to port each package. However, we found such a high variance (400%) in the time to port the first few packages that we thought it not useful. However, as we show in the results section, the metric can be of merit.

Stdlib and Third-Party Dependencies

Besides the dependencies within our own codebase, we also need to determine whether our packages have stdlib or external/third-party dependencies that would prevent us from upgrading.

We leveraged the caniusepython3 [19] tool to check.

To do so, we needed `requirements.txt` files for each project. This was done by installing each project in its own respective virtualenv then running 'pip freeze'. These `requirements.txt` files were then passed to caniusepython3.

This helped us discover a number of third-party packages that we would need to find replacements for.

Workflow

Starting point Initially, we ported the simplest package, with the least dependencies. This allowed for some learning, which we could apply to subsequent packages.

Procedure We created a wiki page with details of the procedure to follow. This was improved over time, and is summarised below. An important detail is that a pull request must be created and merged to the main branch after each step. This helps to ease the burden on the reviewers — smaller changesets are easier to review, and result in better quality reviews.

1. Evaluate the repository. This provides context for the developer — they may have minimal experience with the repository.

2. Clean up. Delete unused code and scripts to simplify the porting effort.

3. Update Continuous Integration (CI) to run python3 tests (allow failure).

4. Futurize module (maintain python2 support):
   (a) Futurize stage 1 (modernize python2 code only, no compatibility with python3)
   (b) Auto code format (using black [20]). Since we are touching the whole codebase, now is a good time to make these kind of large scale changes (code was not previously auto-formatted).
   (c) Futurize stage 2. This uses the python-future wrapper to provide python3 support. We decided to avoid `unicode_literals` due to the drawbacks [21], and the `future.newstr`, as it leads to type confusion — lines like this were removed: `from builtins import str`.

5. Futurize module (add python3 support). Get tests passing under python3. This includes updating dependencies, dealing with strings, and other python2-python3 differences. This is also the time to update the `classifiers` and `python_requires` in the `setup.py` file.

6. Update CI to fail if python3 tests fail, and to publish the python3 package.

7. Verify that downstream packages still pass their python2 unit tests. This ensures no regressions in the python2 version of the ported code.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV036

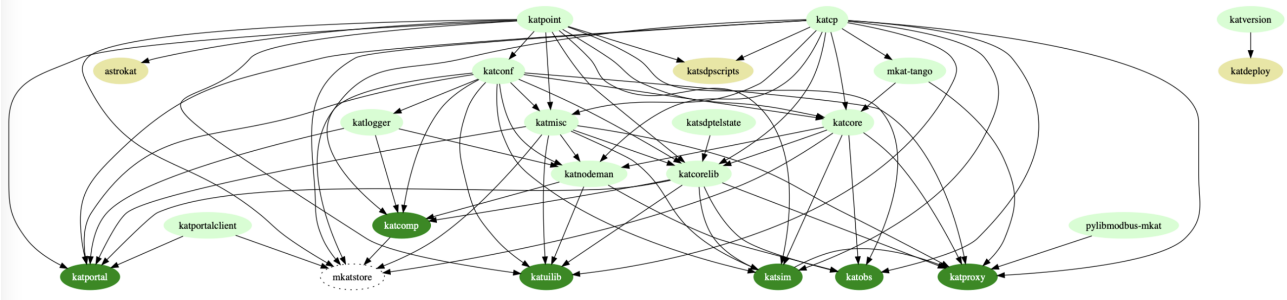


Figure 1: Python package dependency graph. Light and dark green packages are used at run-time, with dark green indicating the leaf nodes. Yellow packages are not used at run-time. White dotted packages are deprecated, and can be ignored.

8. Verify that the full integration tests for the whole codebase are still passing. Again, this is largely running under python2, until we get to the leaf nodes, but increases our confidence in the port.

Finally, once all the packages support python3, we would start dropping python2 support completely. This would enable python3 features like `async` and `await` to be used. It would also be possible to drop the python-future compatibility layer.

Human resources We follow an Scrum approach, with work divided into sprints. Initially, we added stories for porting packages into each sprint. This was a failure for a few reasons:

- It proved impossible to estimate the time to migrate a package. Stories we expected to fit in a sprint, would run to multiple sprints. Even as we re-estimated them each sprint.
- The assumption of "all developers are equally skilled" made by Scrum, meant that different people would work on porting in subsequent sprints. This meant a lot of the knowledge gained in previous sprints had to be learned afresh, by the new developers that had not done this kind of work before. It was very inefficient.

Due to these failures, we decided on an alternative approach: dedicate a single experienced software engineer to the task, with reviews from the technical lead. This work was handled outside of the Scrum process, negating the need for the problematic estimation. A downside was that we were not even able to speculate a completion date. The benefit being that one person would become an expert at the porting techniques and typical problems, allowing for a quicker completion.

The single developer approach has shown some dividends. In practice, the main problem was prioritisation. There were often support issues needing urgent attention that arose in the areas where he/she is most experienced. This removed the focus from the porting work, which while very important, could never have a priority as urgent as keeping production operational.

A slightly more minor problem is the bottleneck introduced by having a single reviewer to sign off on the changes. As the approach guidelines require pull requests to be merged after each step, this can lead to short delays. In practice, we worked around this by starting the next part of work from an unmerged branch. Then if any rework was required on the open pull request, that would be applied to the new branch (e.g., via git rebasing).

In the last 4 months, we have moved a second resource to this work to try and further speed things up.

We do not track the hours that each developer works on any given task, so there is no estimate of the total hours consumed by this effort.

RESULTS

Test Coverage and Rework

If we measure progress as "packages done", then one of the issues that effectively delayed us, was rework. After porting package *A*, testing it and the downstream packages dependent on it, we would declare the package as "done". However, when we started porting package *B*, which depended on package *A*, we would find a defect that needed to be fixed in package *A* again. This was generally because that part of package *A* was not properly covered by test cases. When we started exercising it under python3 in a different way, problems emerged.

This is a reason why good test coverage is very beneficial when undertaking a task such as porting. It can be seen as a type of refactoring - the functionality must remain unchanged, while some improvement is made to the code.

Another area where test coverage is especially important at the leaf nodes in the dependency graph. Here there are no further downstream packages to provide additional testing feedback. If the test coverage is low, then we have little confidence in the port, and must resort to manual testing (exercising the applications by hand). This is a risky strategy.

Metrics

For each package we used our version control system history to extract the dates of the first and last commit related to the porting work. This excludes the rework mentioned in the previous section. Using the time elapsed between the first and last commits and the SLOC, we can derive useful metrics.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV036

The rate (SLOC per week) to port a package as function of the package size (in SLOC) is shown in Fig. 2. We see no significant correlation. The time axis is not shown, but there was a factor of 4 difference in the rate for the first few packages (0.29 to 1.19 kSLOC code per week).

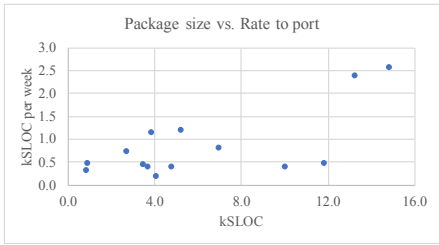


Figure 2: Package size versus rate of porting for all packages completed.

Considering the total SLOC in the codebase, minus the SLOC completed over time we can get a "burndown" chart, as in Fig. 3. There have been significant variations in the rate of work, but overall the trend is linear. Unfortunately, after almost 3 years of part-time work, we are only half way. Extrapolating this line, we predict an end date around January 2025. We need to decide if progressing with this porting project is worthwhile, as there will only be a one or two years of MeerKAT's operational life remaining. Alternatively, we need to find a much more rapid way of progressing.

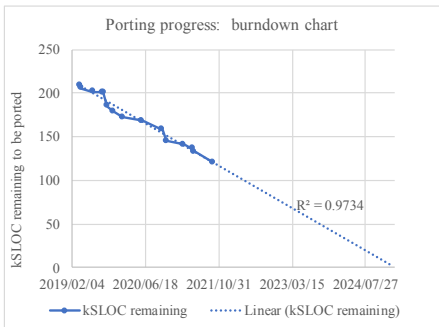


Figure 3: Lines of code remaining to be ported versus time. With completion date projected using a straight line regression.

CONCLUSION

We have discussed our motivation and the various strategies available for porting our codebase from Python 2 to Python 3. We showed how the dependency graph drove the approach, and provided a detailed procedure. The various missteps and challenges associated with human resources and project prioritisation were described.

The results section indicate the disappointing progress we have made thus far. We will need to re-evaluate the porting project to see how we can change our approach. If we had done such close analysis of the data a year or more ago we may have adjusted sooner.

We have not yet worked on legacy and proprietary packages that may be difficult or impossible to port, as these are in an unreached leaf node. Nevertheless, our plan is separate out the parts that are too hard to port, and run them in containers that still have the old dependencies.

The porting of a codebase is a significant undertaking, and needs to be carefully managed. Prioritisation and dedicated resources are important, but success is not guaranteed.

REFERENCES

[1] About MeerKAT - SARAO, https://www.sarao.ac.za/science/meerkat/about-meerkat/

[2] Square Kilometre Array, https://skatelescope.org

[3] Python Programming Language, https://www.python.org

[4] Python 3 Q & A, https://ncoghlan-devs-python-notes.readthedocs.io/en/latest/python3/questions_and_answers.html

[5] Cheat Sheet: Writing Python 2-3 compatible code, http://python-future.org/compatible_idioms.pdf

[6] Python – Debian Wiki, https://wiki.debian.org/Python

[7] Supporting Python 3: An in-depth guide, https://python3porting.com/strategies.html

[8] Python-Modernize, https://python-modernize.readthedocs.io/en/latest/index.html

[9] 2to3 - Automated Python 2 to 3 code translation, https://docs.python.org/3/library/2to3.html

[10] Six: Python 2 and 3 Compatibility Library, https://six.readthedocs.io

[11] Python-Future, https://python-future.org

[12] pydeps, https://github.com/thebjorn/pydeps

[13] snakefood, https://github.com/GreatFruitOmsk/snakefood

[14] pipdeptree, https://github.com/naiquevin/pipdeptree

[15] pip - The Python Package Installer, https://github.com/pypa/pip

[16] Graphviz, https://graphviz.org

[17] pipdeptree patched, https://gist.github.com/sw00/b3a3a4660ef5ff7dcb28a59326f1e9bd

[18] Radon, https://radon.readthedocs.io/en/latest/

[19] caniusepython3, https://github.com/brettcannon/caniusepython3

[20] Black: The Uncompromising Code Formatter, https://github.com/psf/black

[21] "Should I import unicode_literals?", https://python-future.org/unicode_literals.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ALBA CONTROLS SYSTEM SOFTWARE STACK UPGRADE

G. Cuni, F. Becheri, S. Blanch-Torné, C. Falcon-Torres, C. Pascual-Izarra,
Z. Reszela, S. Rubio-Manrique, ALBA-CELLS, Barcelona, Spain

Abstract

ALBA, a 3rd Generation Synchroton Light Source located near Barcelona in Spain, is in operation since 2012. During the last 10 years, the updates of ALBA's Control System were severely limited in order to prevent disruptions of production equipment, at the cost of having to deal with hardware and software obsolescence, elevating the effort of maintenance and enhancements. The construction of the second phase new beamlines accelerated the renewal of the software stack. In order to limit the number of supported platforms we also gradually upgraded the already operational subsystems. We are in the process of switching to the Debian OS, upgrading to the Tango 9 Control System framework including the Tango Archiving System to HDB++, migrating our code to Python 3, and migrating our GUIs to PyQt5 and PyQtGraph, etc. In order to ensure the project quality and to facilitate future upgrades, we try to automate testing, packaging, and configuration management with CI/CD pipelines using, among others, the following tools: pytest, Docker, GitLab-CI and Salt. In this paper, we present our strategy in this project, the current status of different upgrades and we share the lessons learnt.

ALBA CONTROLS SYSTEM DEVELOPMENT AND EARLY OPERATION

When building ALBA controls system, the main goal from the software (and hardware) point of view was to use standard tools and be as homogeneous as possible in order to ease development, training, and troubleshooting. Beamlines and accelerators have the same software structure and use the same applications wherever it is possible, for example, vacuum or motion control.

Controls Software Stack Overview

ALBA controls system uses Tango as middleware, a distributed control system framework based on CORBA [1]. It is characterized by a client-server architecture, its object-oriented design, and the use of the database as a broker and name service. The ALBA Controls Group chose Python as the main programming language and strongly invested in developing and supporting PyTango, a Python binding to C++ Tango library [2].

The vast majority of GUIs at ALBA are developed using Taurus [3], a library for building desktop applications in PyQt. Taurus was initially connecting only to Tango models but over years its architecture evolved towards a highly modular and data source agnostic solution broadly used in numerous scientific installations.

Apart from Taurus, other generic and transversal services, initially implemented as Tango device servers interfaced from GUIs, gradually evolved into projects used not only at ALBA beamlines and accelerators but also at many other institutes of the Tango community. These are, among others:

- Sardana, a scientific SCADA suite [4], which consists of Taurus-based widgets for experiment control and IPython based CLI called Spock on the client-side, and a powerful sequencer called MacroServer and Device Pool for interfacing with the hardware on the server-side.
- Panic, an IEC62682 compliant Alarm Handling suite (Alarm Handling Panic GUI and PyAlarm) [5] capable of messaging and automated execution of control system actions.
- Generic tools and device servers (Tango import/export scrips, calculation device servers, vacuum controllers, diagnostics tools) [6]

Apart from the generic services and user interfaces, every sub-system has its specific applications, e.g. MX-CuBE control application for macromolecular crystallography experiments used at BL13, TXM control application for tomography experiments used at BL09, or the accelerator timing system controls stack (Linux drivers, Tango device servers and GUIs).

The ALBA Controls Group used to manage all the software under maintenance with the “bliss” system. The bliss system, developed by the ESRF, is an rpm-based packaging and Software Configuration Management (SCM) tool. It comprises two applications: the blissbuilder and the blissinstaller, both offering intuitive to “non-packaging experts” graphical way of defining and creating packages, and installing them, at the same time being limited in terms of automatic package creation and deployment.

The different pieces of software run on diskless compact PCI (only for the accelerators) and industrial PCs, distributed in the service area or experimental hall with direct access to the hardware devices. The boot servers, archiving, Tango databases, CCD data acquisition, and various other services run on VMs centralized in the computing room. Most of the controls hosts run a standard Linux distribution which at the beginning was openSUSE but there are also some Windows hosts, mainly workstations for data analysis.

ALBA CONTROLS SYSTEM SOFTWARE OBSOLESCENCE

A control system for a scientific facility such as ALBA is not a static system: new hardware needs to be suppor-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

ted, scientific software introduces requirements for up-to-date libraries and other dependencies, etc. Similarly, the different components of the system are heavily interconnected, which makes that, very often, the upgrade of a component requires similar upgrades in other related components.

While the initial stack of technologies and strategic decisions proved successful in the first stage of building the ALBA controls system and first years of commissioning and operation, they inevitably needed to be reviewed and updated after more than a decade of use and in the new stage of operation.

Operating System

Most of the control machines (and also the generic IT servers and even personal desktop machines) installed during the first years of ALBA were running OpenSuSE 11.1, which had already reached its end-of-life in 2011, even before ALBA entered into operation. This imposed many limitations on the support for new software and hardware, due to being tied to core libraries and modules (e.g. libc, Python, numpy, PyQt4,...) that dated back to 2008. As this situation became increasingly problematic, some selected machines, typically those accessed by end-users, were upgraded to newer SuSE versions (mostly OpenSuSE 12.1 , but also up to OpenSuSE 13.1), which alleviated some specific issues but increased the maintenance effort due to the larger number of platforms being supported.

Operating systems security updates became scarce after some years in operation, we had a big dependence on 32 bits systems that were not already supported with a lack of hardware drivers for evolving these systems into more up-to-date platforms. This situation was not sustainable, and by 2015 a task force was constituted to evaluate the different options for a general upgrade of the OS of control system machines and also on the workflows for simplifying future upgrades. As a result, in 2018 ALBA started replacing OpenSuSE with Debian in the control system machines (see more details in the "ALBA Controls System Software Upgrade" section).

Packaging System

From the beginning, it was decided that all the software should be packaged in order to be deployed on a control system machine. This included software that was directly developed by ALBA's control group but also any other software that needed to be installed and which was not available for our OS as a system package from official or third-party repositories. Furthermore, it was decided to separate the management of the software that was installed as part of the control system from that related to the "general operating system". To that effect, and in order to facilitate the packaging tasks, the "bliss" system from the ESRF was selected and custom-adapted to ALBA's own conventions. While the packages produced and managed by bliss were of the same type (RPMs) as those of the system, they were different in that they were unpackaged into non-standard paths and did not follow the same quality rules (e.g. the dependency declaration was managed by bliss and was very limited in comparison with that of the system packages). This decision made more sense in the early times of ALBA, when the standard package build and management tools from the Linux distributions were much less user friendly, but it introduced several limitations that recommended its reconsideration:

- the separation between "general system" and "control system" was often not clear (e.g. in the case of generic libraries such as hdf5 which were also dependencies of control system packages, or the case of custom kernel drivers necessary for hardware support, or system daemons which were hard to manage by bliss).
- it was difficult to have a clear view of the whole stack of software required for a given purpose since its management was splitted into different tools (standard ones for the general system and bliss for the control system)
- the custom nature of the bliss packages prevented us from collaborating with other facilities in the packaging of software (even for software developed in collaboration, such as e.g. Tango)
- while bliss had a very gentle learning curve, it being a custom solution often required using workarounds to several of its limitations, which were not properly documented but passed as "group-lore"
- with the advent of containerization, the use of a custom packaging system made it very difficult to benefit from standard container images
- the maintenance of the bliss packaging system itself was a burden for the control system group

Because of these reasons, it was decided to move towards using the native package system for the host OS and standard package management tools as part of the change to an updated OS.

Python

Most of the software developed and maintained at ALBA is based on Python. Because of our dependency on old OpenSuse 11.1 we were forced to support Python 2.6 this disincentivated the adoption of Python 3 within the ALBA controls team even after 12 years of its release in 2008. As a consequence, in 2021, after Python 2 reached its end-of-life we still have a considerable part of our software depending on Python 2. This is especially problematic because most of the core modules on which we depend had actively stopped supporting Python 2 and current Linux distributions already stopped providing them, making a gradual transition more difficult and forcing us to keep whole systems outdated because of it.

A lot of effort is being put on adapting our software to Python 3, starting in 2019 when Taurus started supporting it (with a common code base that also supported Python2), and following with Sardana, some GUIs, Tango

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Device Servers, etc. mostly associated to the OS update of machines in the Control Room and Beamlines.

Tango

The development of ALBA Control System was initiated using Tango 5 on OpenSuSE 10.2 platform. Prior to starting operation it already evolved to Tango 6 over OpenSuSE 11.1 and later to Tango 7, enhancing the performance of the system and providing what has been our stable platform for 10 years. Despite of the success on the development of the control system, several limitations generated recurrent issues, especially the dependence on outdated Java packages due to old OS and the lack of performance of the OmniNotify event system (notifd), that limited the event throughput causing continuous incidences and blocking the evolution to a full event-based architecture.

An initial upgrade project was started for several ALBA beamlines, for which a hybrid Tango 7/Tango 8 system was deployed. Tango 8 introduced ZMQ as a new event system, thus increasing the theoretical performance of the control system and eliminating the need for a permanent daemon dedicated to event dispatching, which was a recurrent cause of memory leaks and event bottlenecks. This hybrid system performed poorly, due to the usage of already obsolete operating systems and the (unknown at the time) incompatibility between different versions of ZMQ. Finally, a full Tango 8 based approach was adopted for those upgraded control systems. Thid required the deployment of multiple Jenkins servers in order to automatically recompile all existing Tango Device Servers for every new release of Tango.

Although Tango 8 was finally discarded as an alternative for the main Accelerators Control System (due to the legacy OS 32 bit dependency), all the work done on CI for the Tango 8 upgrade became the base for the current CI/CD workflow, which allowed a much smoother upgrade of both OS and Tango to the latest versions (Tango 9.3.4 over Debian 9), which solved all the previous performance issues and allowed a successful migration of all control systems at ALBA. The experience during the migration process also triggered the Tango Community to move from short-term releases to Long Term Supported (LTS) releases, a move that benefited the whole community in terms of development stability and larger reusability of software stack throughout the community.

PyQt and PyQwt

Taurus, and as a consequence most GUIs at ALBA, was originally based on PyQt4 [7] for its general widgets and on PyQwt5 [8] for its plot widgets. PyQt4 reached its end of life in 2018 and PyQwt5 was by then already unmaintained (its latest release from 2011) and was never ported to PyQt5 or Python 3.

As with Python, the need to run the control GUIs on an obsolete OS shipping outdated versions of PyQt slowed the porting of our own software. Taurus started supporting PyQt5 in 2019 and switched to it as its default binding in 2020. It also started implementing alternative plotting widgets based on PyQtgraph [9] in 2017 but these only became ready to replace the old PyQwt5 ones in 2020.

After that, some GUIs (mostly the most modern ones which were already based on Taurus >= 4) could be ported with little effort, but many remain unported because they still use Qt APIs that were already deprecated in 2013. These are being gradually migrated as part of the effort to update the OS of machines in the Control Room and Beamlines.

ALBA CONTROLS SYSTEM SOFTWARE UPGRADE

New Operating System Selection

In 2015 a task force involving various members of the controls group in coordination with the IT infrastructure group started evaluating the various candidates to replace the old OpenSuSE 11 and 12 machines. One key decision was to decouple the choice of the OS for control system machines from that for generic IT systems because it became evident that, apart from a common requirement for robustness and long term maintainability, the requirements were not compatible: in the case of the control system we stressed the availability of scientific software and the support of scientific instrumentation and varied hardware platforms, while for generic IT systems the focus was on vendor support and network and filesystem robustness. As a result, OpenSuSE Leap and Debian were shortlisted as the candidates for the control system, while the IT systems moved towards CentOS.

During two years we conducted an in-depth evaluation of OpenSuSE (using versions 13.1 and 42.1) and Debian (with versions 8 and 9). The main aspects being evaluated were:

- life cycle and duration of supported updates
- amount of available software relevant for ALBA (i.e., mostly science-related). In order of preference we considered packages already included in the distribution, then those officially packaged for the distribution by its authors and then those packaged unofficially by some third-party
- usage in other similar facilities, specially within the Tango Collaboration
- relevant hardware platforms supported. In order of priority: amd64, i386 and arm
- quality of the documentation and community support
- quality of native package build and management tools

These aspects were not just evaluated from theoretical research, but also tested in practice by deploying both systems during more than a year to production environments (a relatively simple beamline and a support lab) for which we had to package and deploy a full basic stack of control system software, ranging from hardware kernel drivers to taurus-based GUIs and including also a Tango system with several device servers.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

This selection process has been constrained by our needs to evolve our stack of technologies, the feasibility of porting existing devices, drivers, and applications from 32 to 64 bits hardware keeping consistency between the new and legacy control systems as well as the necessity to keep our CI/CD workflows in-house to avoid dependency from external services. To be able to recompile and test all the software stack onto new platforms, multiple Jenkins servers were configured for all our existing OS at ALBA (SuSE 11/12/42 and Debian 8/9 in 32/64 bits), including physical hardware machines to test and evaluate hardware drivers compatibility and performance. It allowed to test all platforms and confirmed the necessity to keep 32 bits support for the next 5 years.

Both distributions showed that they were fit for the task, but Debian performed a bit better in most of the aspects being evaluated and had better support to all our hardware platforms, to the point that it was enough to overcome the main advantage of OpenSuSE which was that it involved less change with respect to the old system. As a consequence, Debian was chosen and, in 2018 the migration started with the upgrade of the Control Room machines to Debian 9, and followed by the machines dedicated to Sardana in the beamlines (at the moment of writing all but two). Once this is done, other systems will be gradually updated, leaving only some legacy systems out of the upgrade.

Packaging

As mentioned in the "ALBA Controls System Software Obsolescence" section, an important conclusion for the OS update was to also switch from our custom packaging system (bliss) to using standard system tools for both building and managing the control system packages.

As part of the new OS evaluation, we evaluated the "OpenSuse Build Service" [10] and the Debian packaging toolchain. Of both systems, we valued that the packaging could be managed with a workflow involving a version control system: a domain-specific one inspired in CVS in the case of OpenSuSE or git in the case of Debian [11].

Once Debian was chosen as our new OS for controls, we produced a step-by-step guide, a set of examples, and a Docker image with the whole Debian packaging toolchain in order to facilitate the transition from bliss to the standard Debian packaging workflow. We provided training to the whole group of controls developers, including hand-on sessions in which we collectively packaged the whole software stack required for the upgrading the Control Room machines.

On a first stage, the packaging workflow was done manually using the mentioned Docker image but, with the introduction of "salsa" service by Debian [12], we extended the Debian CI pipeline to automate the process [13]. Currently, the whole packaging process is done by a Gitlab CI pipeline triggered by pushing a version tag to the packaging repository and resulting in the package being automatically built for various distributions, tested and uploaded to a staging package repository. User intervention is required only for adjusting the package configuration during the creation of the first version of the package, or when the package configuration needs to be modified (e.g., for updating a dependency). These interventions can be entirely done using the web interface of Gitlab, eliminating the need of locally installing the packaging toolchain. The same toolchain is also capable of promoting the package to a production repository if the user confirms that it is ready after testing it using the staging repository.

The packages produced using this pipeline need to pass the same automated quality assurance tests used for the official Debian packages, resulting in much better quality packages than those that were created with bliss.

GUIs for Accelerators Operation

The upgrade of Graphical User Interfaces (GUI) for operation has been a critical procedure, as it has been performed gradually without interrupting the operation of the accelerator. A gradual replacement of Control Room consoles has been performed over a 2 years period, in which all legacy OpenSuSE machines have been replaced by Debian, upgrading from Taurus 3 to Taurus 4 and Tango 9 in the process.

Paradoxically, the performance increase provided by Tango 9 caused several issues on old applications, due to the higher event throughput on those applications not properly designed to manage high update rates. Replacement of old applications by newer versions was performed on various stages, following a strict schedule in which users were informed and allowed to test each application prior to its deployment in production. Several applications required changes in its management of attribute updates. Having all applications built on top of a common framework (Taurus) helped to implement the solutions required on Taurus itself, allowing the upgrade of multiple applications at once and allowing to solve issues once-for-all as they appeared providing flexibility at the application level to choose between polling or event-based refresh for each attribute.

In addition to the Tango upgrade, several other technologies were upgraded. Qt4 and Qwt libraries were replaced by Qt5 and PyQtGraph, and the database backend for both Tango database and Archiving System was upgraded from MySQL 5.0 to MariaDB 10.0 the newest event-based Tango Archiving System [14]. The adoption of Taurus 4 on Qt5, which was profoundly refactored, required to modify existing applications to use new Qt style signals, and the migration to PyQtGraph of archiving visualization tools is still an ongoing project, for which new Python 3 database extractors have been developed capable of online decimatiob of the incoming data from the high-performance event system.

The usage of Taurus framework for developing all custom applications required for day-to-day operation allowed to do a transparent upgrade of those applications developed using only generic Taurus widgets[3][15], allowing to upgrade dozens of applications without only minor modifications, mostly on unit visualization

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

(managed by Pint on Taurus 4) and attribute values formatting.

Tango Device Servers Upgrade

Upgrading device servers to new technologies have become one of the most difficult processes, due to the dependency on old hardware and the lack of drivers for newer operating systems. Industrial PC's devoted to hardware testing have been deployed on several ALBA laboratories for testing every upgraded device server prior to its deployment in production. Drivers have been updated whenever open source code was available, and several projects have been started to migrate DAQ acquisition cards to newer platforms.

In the meantime, most of the hardware-based control systems are still kept on legacy control system, while only ethernet-based hardware (PLCs [16], cameras, scopes) have been migrated to newer platforms using virtualized hosts. We are in the process of migrating all serial-line based device servers using serial-to-ethernet adapters, thus eliminating the need for Industrial PC's for most vacuum systems, and a new 32 bit Debian 9 image is under development to replace most of our legacy control systems. We expect to migrate two thirds of our legacy control system to Debian during the next year, but assuming that still many hardware dedicated IOCs will have to be kept on legacy versions waiting for a future hardware replacement.

For those devices already running on Debian 9 and all our purely software-based devices (database extractors, data processors, calculation and simulation devices, alarm systems) we started a project to migrate them to Python 3. The project started with the migration of the common-shared libraries (fandango [17], panic [5], pytangoarchiving) and the adoption of the new Python HL Tango API for every new development, which provides fully Python 3 compatible code.

Sardana Python 3

Sardana is used by various European institutes, mainly synchrotrons, and at ALBA it is used at the beamlines, accelerator, and auxiliary laboratories. Due to the software stack obsolescence at ALBA, the Sardana codebase was required to maintain compatibility with Python 2.6, Tango 7, PyQt4, etc. for a long time. Even if the Sardana community was well aware of the Python 2 EoL the Sardana codebase migration to Python 3 was always being delayed in favor of other developments. This changed when the ESRF announced that after its accelerator upgrade program they would like to use Sardana with Python 3.

Sardana is a final application, not a library, and it followed a different migration approach than the Taurus project. Sardana codebase was migrated to Python 3 without supporting Python 2 at the same time. Sardana users highly demand new feature requests and bug fixes that are continuously being released. So, it was in the interest of both users and developers that the setups were upgraded to use the new Python 3 compatible version ASAP. The upgrade process required the end users to migrate their Sardana plugins to Python 3 as well.

In the case of ALBA, the Sardana upgrade to Python 3 required upgrading the OS to Debian 9, Tango to version 9, and Taurus to version 4. The project scope was limited to upgrade only the strictly necessary part of the controls system to provide to the users the Sardana service running with Python 3. As a consequence, the coexistence of different software versions in different sub-systems was assumed. A transversal workforce of nine controls engineers was formed to work on the software migration to Python 3, packaging, testing, and commissioning.

The upgrade process consisted of two phases, first, the Tango database service was upgraded to Tango 9. In this process, the TANGO_HOST configuration was changed to use a DNS network alias, in order to facilitate eventual rollbacks and future upgrades. Here, two problems appeared: the Tango 7 event system started to suffer memory leaks in certain conditions and the Tango Java event system stopped working. In the second phase, the Sardana Tango servers were moved to dedicated VMs and a new workstation was prepare to run the Sardana client applications. Here, appeared problems with the Tango 9 causing deadlocks and hangs of the experiment procedures.

The upgrade process required from the users a thorough testing process, nevertheless, rollbacks were not avoided. In order to advance in the upgrade, immediate workarounds were applied and issues were escalated to the Tango community, which was always very helpful in debugging and solving problems. In the current state, all, but two ALBA first phase beamlines were successfully migrated and all the second phase beamlines were built using Sardana Python 3. Other institutes in the Sardana community are also very well advanced in the upgrade process.

Following Debian updates

The selection of Debian as a new OS for the control system was never considered as a one-time upgrade effort but was more a commitment to try to follow future Debian releases according to the needs and capabilities of the ALBA Controls Group. New Debian releases happen every two years, are supported by three years by the Debian organization, and later receives an additional two years of Long Term Support. During the three years of support, the release is updated once in a while with security fixes and fixes for important bugs. Those updates are called "point releases".

All the control system Debian hosts at ALBA, unless explicitly excluded, are upgraded to point releases during short shutdowns (several times per year). This upgrades only the non-control-system packages and the whole process is highly automated using the Software Configuration Management tool.

Due to the still not fully automated packaging creation for the current and future Debian releases, the upgrade to Debian 10 could not start as soon as it was available. Since the packaging infrastructure is now ready, soon we will start a project of upgrading the Sardana service to use De-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

bian 10, which will open new possibilities in the project development thanks to the updated dependency stack. This will be a minor effort thanks to the automatic packaging and isolation of the Sardana Tango device servers on a dedicated VM.

Debian 11 was released in August 2021 and upgrading to it instead of Debian 10 has been discussed and is still an option, but the decision is strongly conditioned by the fact that Debian 11 does not provide a full Python 2 stack nor PyQt4 and therefore all software running on it needs to be ported to Python 3 and PyQt5 (or alternatively, run on virtual environments or containers).

APPLIED PRACTICES AND TOOLS

Each author should submit the PDF file and all source files (text and figures) to enable the paper to be reconstructed if there are processing difficulties.

Testing

Testing plays a very important role in the control system upgrade process, and testing strategy must be tailored to the nature and development status of each of the projects.

The SEP5 [18] introduced the first conventions for developing automatic tests with unittest module (part of the Python standard library) for Sardana and Taurus on the unit and integration level. Setting up Continuous Integration (CI) jobs for running those tests provided valuable feedback and let us avoid many new bugs and regressions before the upgrades in the production setups. Since the testing coverage is still not satisfactory manual tests are performed before every major release. Recently we decided to switch from unittest to PyTest [19] for developing automatic tests because this second one provides a better programming interface and many useful features available out-of-the-box e.g. fixtures, parameterization of tests, code patching, temporary resources context managers.

Developing automatic tests for the Tango control system requires setting up necessary resources for the needs of the tests. Our initial approach was to set up a disposable container running a Tango database with prepared instances of the Tango devices and starting and stopping Tango device servers for different classes of tests. Recently, in some tests, we started employing a different approach and use the PyTango (Multi)DeviceTestContext which allows better test control and isolation.

Automatic tests for the controls software can either interface with the real hardware or with simulators [20]. Currently, we employ this second strategy, for example, in the case of Sardana all the possible controller types e.g. motors or experimental channels are available in a simulation mode and are used during the tests. Similarly, in the case of the PyIcePAP, a Python library for interfacing the IcePAP motion control, simulated hardware is used during the tests. Furthermore, in the near future, we plan to employ automatic tests with the real hardware equipment located in our computing laboratory into our CI and in addition run nightly stress tests to discover non-easily reproducible bugs and performance degradations.

Automation and Reproducibility

The advantages of using Continuous Integration for the testing of both Taurus and Sardana have already been described in the previous section. Our experience in these cases has been unequivocally positive, because the use of CI not only improved the overall quality of the code but also enabled a much more agile collaboration as a consequence of the reduced risk of regressions. We will certainly promote the use of CI in other developments.

The use of Continuous Integration for software packaging in ALBA has also been discussed above. It is important to note that our current pipeline enables going one step further towards using a Continuous Delivery approach. Also, the fact that the packages are now artifacts of a fully reproducible pipeline whose logs are always available for inspection, has also facilitated the collaboration and support in the packaging process.

Additionally to the use of CI/CD for the packaging of our internal developments, we are also using it for publishing some of our software to PyPI, to Conda-Forge and other Anaconda Cloud channels, and even to salsa for inclusion in the official Debian distribution. This greatly improves the visibility and usability of our code outside ALBA.

Finally, it is also interesting to mention another area in which we found automation to be very helpful: the generation of documentation for our projects. In the case of Taurus and Sardana, we originally started by generating the documentation of the Taurus and Sardana projects using sphinx, and soon we automated it thanks to the ReadTheDocs service which we found excellent. However, we eventually decided to move the documentation build to the same CI system that we used for the code (first, Travis CI and now Gitlab CI [21]) in order to unify the CI. Apart from these projects, we also use Gitlab CI for autogenerating documentation of internal software and deploy it automatically.

Configuration

The ALBA Controls Group uses Salt [17] to configure and manage software on remote nodes. We are moving to the central Salt Master the following tasks: the installation of Debian packages, the clones of Git repositories, the installations with Pip, and the installation and configuration of the Conda environments. We define a catalog of ALBA Services e.g.: Tango, Sardana, Taurus, Icepapcms, Archiving, etc. The application of the Salt recipes allows us to automate the installation and configuration of the same Services in parallel in different machines and in a reproducible way. Each remote node is configured through Salt Grains that specify which Services should be installed in that node.

We are still in the process to fully adapt the Salt solution to ALBA controls system and we have some open points, like the best use of version control of the services,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the proper workflow for testing a new version, or the use of a graphical user interface.

One particularly problematic point is how to manage rollbacks: the Salt recipes express the changes to be done in the target machine, which is not exactly equivalent to specify the state of the system. It is easy to conceive a situation in which reverting a salt change leaves the system in a state which is not identical to that previous to the original application. For the moment, the use of "snapshots" in the case of Virtual Machines seems the best workaround.

Control Room Scheduled Updates

The upgrade of existing applications at ALBA Control Room required a careful approach, as it needed to guarantee maximum availability and stability for the operation of ALBA Accelerators. It required careful testing of every application, including hardware interaction, before applying any update to the main operator consoles. The need for hardware tests also forced us to integrate the testing procedures with the Accelerators Operation calendar, thus constraining the CI/CD workflow and defining our Control Room Update Flow states:

- Testing prior to deployment: Developers perform tests of GUI applications and device servers under development (from git or staging repositories) on a dedicated machine, only during machine maintenance periods (8-9 dedicated weeks per year).
- Testing on production: Packages are deployed on a machine available to machine operators that can be used to validate the latest releases (staging) of each software package prior to its propagation to the Control Room operators consoles. Applications are kept on this stage for at least a machine-run period (4 to 5 weeks) prior to their promotion to the production environment (only after validation by users).
- Production: Finally, validated applications available in the production repository are deployed into the 10 operator consoles in the Control Room for its use by all operators. Prior to the update, all previous stable versions of packages are deployed into an "old_stable" machine, available as a backup in case a bug appears on operation despite all the previous testing.

This workflow implies a delay of 5-6 weeks before a development reaches its production status, a conservative approach in order to guarantee the maximum stability of the system. In the case hot-fixes are required, a special mechanism to perform updates during machine days is enabled (once per week), which requires a notification to affected users via an Accelerator Interventions ticketing system and an update of the configuration system (Salt recipes), which is managed by a git repository thus providing a register of changes.

Containerization

Up to now our use of (Docker) containers has been mainly limited to the context of software development as, e.g.:

- providing the environments for the CI jobs
- providing a pre-configured a clean environment for manual testing or packaging of software
- developing or debugging in a reproducible environment

However, we are also considering the use of containers for deploying the control system in ALBA. In particular, we are currently evaluating how to deploy our typical beamline stack of applications as microservices in a Kubernetes cluster. Some of the expected potential advantages of this approach are:

- it could facilitate upgrades, thanks to the increased isolation of each component
- rollbacks would be much easier since the whole system can be re-deployed from scratch from a version-controlled configuration
- it would simplify the debugging thanks to the possibility of running clones of a given deployment

TOWARDS ALBA II

The recently approved ALBA II project will consist of an upgrade of our installation to the 4th generation class of synchrotron light source and is planned to happen in 2028. The new accelerator will provide lower emittance and higher brilliance beam to the beamlines and new challenges to the control system. The ALBA Computing Division started preparing for the future requirements by analyzing the lessons learnt from the ALBA construction and operation as well as identifying and overviewing cutting-edge solutions at similar facilities. In the following years, it is planned to start exploratory projects to acquire the necessary expertise, at least, in the following mainstream technologies which if applied could ensure long-term maintainability of the ALBA II control system. First, isolation of the controls software stack from the host OS, by use of containers or isolated environments e.g. conda. Second, use of web technologies for operators GUIs, which in comparison to desktop applications are cross-platform compatible and more manageable (isolation is done on the server placing minimal requirements on the end-user workstation). We also look forward to the upcoming Tango 10 version release, which code will be refactored/rewritten in order to make it immune to the obsolescence of libraries and technologies e.g. CORBA.

In parallel, some more specific projects for the ALBA II control system have already been started. Here we would like to mention a joint effort between BESSY, DESY, and ALBA in the development of the third harmonic cavity with the DLLRF at 1.5GHz. From the side of controls, accelerators provides us with interface information to the FPGA they program. We have developed a Python binding using cython to access this FPGA and the rest of the cards of the MicroTCA used for

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV037

this system. This binding is used to simplify their development as well as to prepare an Agent in the DCS to represent this card. The interface of registers provided by accelerators is described in the csv file and this file is used as a source for an auto-generation tool capable to build not only the Tango device server but also the Taurus GUI, based on conda environments using Python 3.9. The current development is installed in an autonomous rack to be moved to even another facility to be used. In the installation, there is a pair of containers with the Tango frontend and the MariaDB backend, and the Tango control and the Taurus GUI work on their own conda environments.

CONCLUSIONS

Decisions from the ALBA control system development, commissioning and early operation were successful in terms of the selected technologies e.g.: Tango, Python, PyQt, Tango, etc. From the perspective of time, we believe that it is crucial to keep the OS updated. In our case, the lack of update in the OS conditioned many other updates and, in the long term, generated more efforts in workarounds than the effort of keeping it up to date. Also, it generated a huge effort when it had to be finally updated. The big turnover in the team composition when transitioning from the development into the operation phase was probably decisive in postponing this effort.

Keeping the controls system up-to-date is a collective responsibility of the whole team. Decisions at different levels, from day-to-day to strategic, should be taken considering the long-term maintainability of the control system. Last but not least, it is highly recommended to continuously follow and explore emerging technologies in order to propose improvements, feel self-confident and determined in proposing and conducting upgrade projects.

ACKNOWLEDGMENTS

All the work presented in this paper is an effort of the ALBA Control Seccion members. We would like to mention here: Jordi Andreu, Manuel Broseta [on leave], Tiago Coutinho [on leave], Roberto Homs, Gabriel Jover [on leave], Jairo Moldes, Daniel Roldan [on leave], and Marc Rosanes [on leave]. The upgrade projects would not be possible without support from other sections of the Computing Division, mainly IT Systems Section, here we would like to acknowledge Sergi Puso for his support in multiple fields and Marc Rodriguez [on leave] for sharing his expertise with us about Salt, and the MIS Section, here we would like to acknowledge Daniel Sanchez for his support in the GitLab administration. The ALBA Accelerators and Beamlines Divisions, as users of the control system, played an important role in the controls system upgrade projects by performing acceptance tests and provided valuable feedback. We would like to recognize the help provided by the Tango Controls Community members (a list of whom would be too large to fit into this paper). Finally, we would like to acknowledge the help from the Debian Science Team, and especially Frederic Picca in guiding us in the process of adopting the Debian packaging policies.

REFERENCES

[1] Tango, https://tango-controls.org
[2] PyTango, https://pytango.readthedocs.io
[3] C. Pascual-Izarra *et al.*, "Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1138-1142. doi:10.18429/JACoW-ICALEPCS2015-THHC3003
[4] T. M. Coutinho *et al.*, "Sardana: The Software for Building SCADAS in Scientific Environments", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper WEAAUST01, pp. 607-609.
[5] S. Rubio-Manrique, G. Cuni, D. Fernandez-Carreiras, and G. Scalamera, "PANIC and the Evolution of Tango Alarm Handlers", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 170-175. doi:10.18429/JACoW-ICALEPCS2017-TUBPL03
[6] S. Rubio-Manrique, T. Coutinho, R. Suñé, and E. T. Taurel, "Dynamic Attributes and Other Functional Flexibilities of PyTango", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper THP079, pp. 824-826.
[7] PyQt, https://www.riverbankcomputing.com/software/pyqt/
[8] PyQwt5, http://pyqwt.sourceforge.net/
[9] PyQtGraph, https://www.pyqtgraph.org/
[10] OBS, https://build.opensuse.org/
[11] gbp, https://wiki.debian.org/PackagingWithGit
[12] salsa, https://salsa.debian.org
[13] C. Pascual-Izarra et al., "Collaborative and Automated Packaging", 32nd Tango Meeting, Prague, https://indico.eli-beams.eu/event/310/contributions/738/attachments/547/700/CollabPkg.pdf
[14] L. Pivetta *et al.*, "New Developments for the HDB++ TANGO Archiving System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 801-805. doi:10.18429/JACoW-ICALEPCS2017-TUPHA166
[15] S. Rubio et al., , "Unifying All TANGO Control Services in a Customizable Graphical User Interface", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1052-1055. doi:10.18429/JACoW-ICALEPCS2015-WEPGF148
[16] PyPLC, https://gitlab.com/tango-controls/PyPLC
[17] Fandango, https://gitlab.com/tango-controls/fandango
[18] SEP5, https://github.com/sardana-org/sardana/blob/develop/doc/source/sep/SEP5.md
[19] PyTest, pytest.org
[20] S. Rubio-Manrique *et al.*, "Reproduce Anything, Anywhere: A Generic Simulation Suite for Tango Control Systems", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 280-284. doi:10.18429/JACoW-ICALEPCS2017-TUDPL01
[21] A. Götz et al., , "Migration of Tango Controls Source Code Repositories", presented at the ICALEPCS'21, Shanghai, China, Oct. 2021, paper MOPV034, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

UCAP: A FRAMEWORK FOR ACCELERATOR CONTROLS DATA PROCESSING @ CERN

L. Cseppentő*, M. Büttner, CERN, Geneva, Switzerland

Abstract

The Unified Controls Acquisition and Processing (UCAP) framework provides a means to facilitate and streamline data processing in the CERN Accelerator Control System. UCAP's generic structure is capable of tackling classic "Acquisition - Transformation - Publishing/Presentation" use cases, ranging from simple aggregations to complex machine reports and pre-processing of software interlock conditions.

In addition to enabling end-users to develop data transformations in Java or Python and maximising integration with other controls sub-systems, UCAP puts an emphasis on offering self-service capabilities for deployment, operation and monitoring. This ensures that accelerator operators and equipment experts can focus on developing domain-specific transformation algorithms, without having to pay attention to typical IT tasks, such as process management and system monitoring.

UCAP is already used by Linac4, PSB and SPS operations and will be used by most CERN accelerators, including LHC by the end of 2021.

This contribution presents the UCAP framework and gives an insight into how we have productively combined modern agile development with conservative technical choices.

INTRODUCTION

The Unified Controls Acquisition and Processing (UCAP) Framework is a recent product in the CERN Controls Software & Services (CSS) group's portfolio. This generic, self-service online Controls data processing platform, enables clients to easily implement and run Controls Device data acquisition and processing in Java or Python.

The main objective of the project is to provide a common approach to solve problems where:

1. data is acquired and grouped from several sources (*Acquisition*),
2. based on these inputs, and optionally internal state, a result is calculated (*Transformation*),
3. the result is made available to clients (*Publishing/Presentation*).

Such "Acquisition - Transformation - Publishing/Presentation" problems are regularly featured in many Controls products, such as data concentrators, software interlocks, logging adaptations and autopilot-style Controls software.

At CERN, Controls software development responsibilities are often split between the CSS group (mainly computing engineers providing frameworks and generic services), and equipment and operations groups (typically experts in their specialised domains). Experience showed that in order to effectively tackle certain problems, collaboration of teams with different backgrounds is essential, as the expertise of Controls framework libraries and accelerator domain knowledge are distributed. The UCAP service aims to streamline development and maintenance by splitting responsibilities:

- the UCAP team provides infrastructure, tools, training and support for development, testing and deployment of transformations,
- while the transformation code and configuration stays under the responsibility of the end-user (typically a domain expert).

This *self-service* model enables end-users to focus on realising their business logic, while the service takes care of secondary tasks, such as process management and monitoring. It is also *scalable*: as service usage increases, only new machines and UCAP nodes need to be added to the system.

Nevertheless, this set up brings other challenges, such as isolating user groups on shared hardware, ensuring dynamic server-side code loading, providing substantial documentation and user-friendly access. At this scale, a high level of *automation* is essential

The "UCAP-idea" originates from 2016, with a first prototype started in 2018. During this phase, the first use case was satisfied, providing transformations on approximately 1000 data streams. The following year, in the beginning of Long Shutdown 2, development started on the operational product – in close collaboration with stakeholders. A few months later, still in early 2019, the system went to production while iterative development continued. As of 2021 Q3, the UCAP service is composed of 105 nodes (isolated deployment units), running on 6 physical servers, performing around 20 000 data transformations.

UCAP IN A NUTSHELL

The UCAP service is a *multitenant* system. *Nodes* are assigned to client groups or use cases for isolation purposes. All nodes are fully functional data processing services, differing only in basic configuration parameters, such as name, description and responsible. The layered architecture, presented in Fig. 1, mirrors the "Acquisition - Transformation - Publishing/Presentation" chain – each layer being responsible for dealing with one aspect of data processing. UCAP complies with this model on the architecture level, avoiding the introduction of any new structural elements in the data representation.

The CERN Control System uses the *Device-Property model* [1], meaning that all endpoints providing data are

* lajos.cseppento@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

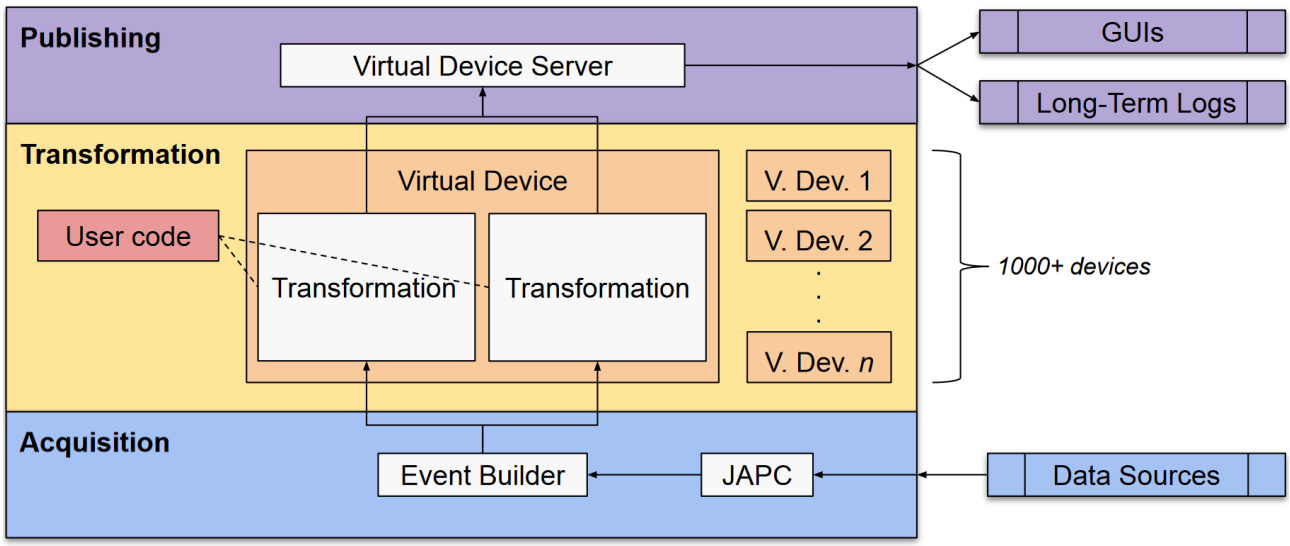


Figure 1: UCAP Node Architecture.

expressed in a `DEVICE_NAME/PropertyName` format, to which data is published as a map of key-value pairs.

Acquisition Layer

This layer is responsible for *acquiring inputs* from data sources. This is carried out using JAPC[1] [2], a CERN standard library for reading inputs from individual Controls Devices. As JAPC is an abstraction layer over communication protocols, a wide range of inputs are supported, including UCAP itself (i.e., chaining of transformations).

In many cases, it is required to group inputs from several Controls Devices, for which *UCAP Event Builders* are provided. These data acquisition algorithms are the driving force for transformations. At the time of writing, 10 event builder implementations are provided, ready to be selected and configured by end-users.

A basic example of an event builder definition is shown in Listing 1, expressed in JSON format.

Listing 1: Basic Event Builder Definition

```
{
  "type":
    "GroupTriggeredCycleStampGrouped",
  "triggerGroup": {
    "subscriptions": [
      { "parameter":"MAGNET_1/Acq" },
      { "parameter":"MAGNET_2/Acq" }
    ],
    "timeoutMs": 1000
  }
}
```

This code snippet instructs UCAP to instantiate an event builder, grouping data from two sources based on their time stamps. The `type` field identifies the name of the acquisition logic, which also decides the rest of the configuration layout.

[1] Java API for Parameter Controls

In this case the `triggerGroup` declares how event creation shall be handled: the `subscriptions` list contains the name of the inputs and the `timeoutMs` field specifies how long to wait for them – this is notably useful when one of the sources is not available, since processing can move with the inputs that were available when the timeout was reached.

Community feedback highlighted how powerful event builders are. End-users can express acquisition logic in a declarative manner, instead of they themselves having to create subscription handles, to start monitoring threads and to implement combination logic.

Each event created contains the data received from all sources and it is passed to the Transformation Layer for processing.

Transformation Layer

The Transformation Layer is responsible for carrying out the actual data processing. Transformations are instances of *converter* algorithms, which are configurable and can be implemented in Java or Python (Python with C/C++ bindings is also supported and used by the community). UCAP currently provides and maintains 20+ so called *standard converters*, which implement commonly required algorithms in a generic manner. Examples include merging homogeneous data from several sources; splitting multiplexed data prior to long-term logging. This enables a part of our user community to benefit from UCAP without having to write any code.

In addition, *custom converters* are also supported. For each event created, a transformation may create one or more results or choose to skip the input. Plus, if there is a need for a stateful transformation, the internal state of the transformation is preserved between events. Apart from these aspects, it is up to the end-user's creativity how they implement their business logic. An example of a Python converter structure can be seen in Listing 2: this particular code re-publishes the sum of two incoming values.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Listing 2: Example Python Converter Structure

```
def convert(event):
    log = logging.getLogger(__name__)

    result = AcquiredParameterValue(
        published_parameter_name,
        event.trigger_value.header
    )

    v1 = event.get_plain_value('BLM1')
    v2 = event.get_plain_value('BLM2')

    result.update_value(
        "result",
        v1 + v2
    )

    log.info("Input : %s", event)
    log.info("Output: %s", result)

    return FailSafeParameterValue(
        result
    )
```

For converter development we recommend using our in-house standards, CBNG [3] for Java and Acc-Py [4] for Python, however, it is also possible without those.

Publishing Layer

In order to make the transformation results visible to consumers, UCAP nodes encapsulate an RDA3[2] Device Server [5] [6] (the same communication protocol used by FESA[3] [7]). The user interface of the Publishing Layer is a simple `return` statement. Internally, the UCAP RDA3 Virtual Device Server takes care of managing client subscriptions. The results can be consumed with C++, Java and Python clients. Results can also be directly logged in the NXCALS[4] [8] time-series data logging system.

Other Notable Features

Apart from "doing the job", UCAP provides several other Controls features, notably:

- FESA-style alarms, which can be integrated e. g., with LASER[5] [9],
- access to previously logged data from NXCALS,
- transparent rolling updates of user code without data loss,
- language-agnostic, RBAC[6]-protected [10] REST API for inspection and management and
- development, testing, monitoring and troubleshooting utilities.

USER EXPERIENCE

User experience is crucial to ensure customer productivity and satisfaction. Good user experience also reduces the support load of the service provider team.

To facilitate development, all Java public API are annotated with javadoc and all UCAP Python public API are decorated with docstrings and MyPy type hints. Sample definitions and converter code are provided in both languages. UCAP includes testing utilities to aid users in writing unit test for converters, inspecting event builders or even running UCAP nodes locally either, directly from Java or using the UCAP Docker image.

When a user wants to start using UCAP, the UCAP team prepares and assigns two nodes for the use case:

- a PRO node for operational purposes,
- a TEST node for verification prior to deployment of transformations in operation.

A rich command line tool, *UCAP-CLI* (see Fig. 2), is provided to facilitate node management and introspection. It has an extensive `TAB`-based auto-completion feature, which aids users with displaying commands, Device, transformation and package names on pressing the `TAB` key. The CLI is also used extensively by the UCAP team for maintenance and support purposes. UCAP-CLI is available on all developer and control room machines, and it can connect to PRO, TEST and local UCAP nodes. The development and maintenance costs of this tool are also low compared to a fully featured GUI.

Out-of-the-box, service monitoring is provided to end-users in the form of Grafana dashboards [11] (example in Fig. 3). These dashboards are automatically created for each node and Device added to the UCAP service, and include subscription (data input) status, conversion errors and input/output rates. These dashboards are quite convenient as in case of failures (e. g., control room display is blank) they make *fault isolation* quick (e .g., inputs are down).

TRAINING AND SUPPORT

Training for end-users is an essential element of the UCAP service. In the early stages of the service, focus was given to close stakeholder collaboration, frequent pair-programming sessions and an emphasis on written documentation. As the user base expanded, effort was put into public presentations (training lectures and product demos) which were recorded and made available online. Since publishing such materials, a correlation with a drop in the number of support requests from new users has been observed.

Communication channels comprise a traditional e-mail list and a Mattermost Channel for instant messaging. Mattermost has shown to be efficient in tackling quick questions. These channels are completed with periodic UCAP Technical Meetings which include demos and hands-on training.

[2] Remote Device Access
[3] Front-End Software Architecture
[4] Next CERN's Accelerator Logging Service
[5] LHC Alarm SERvice
[6] Role-Based Access Control

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

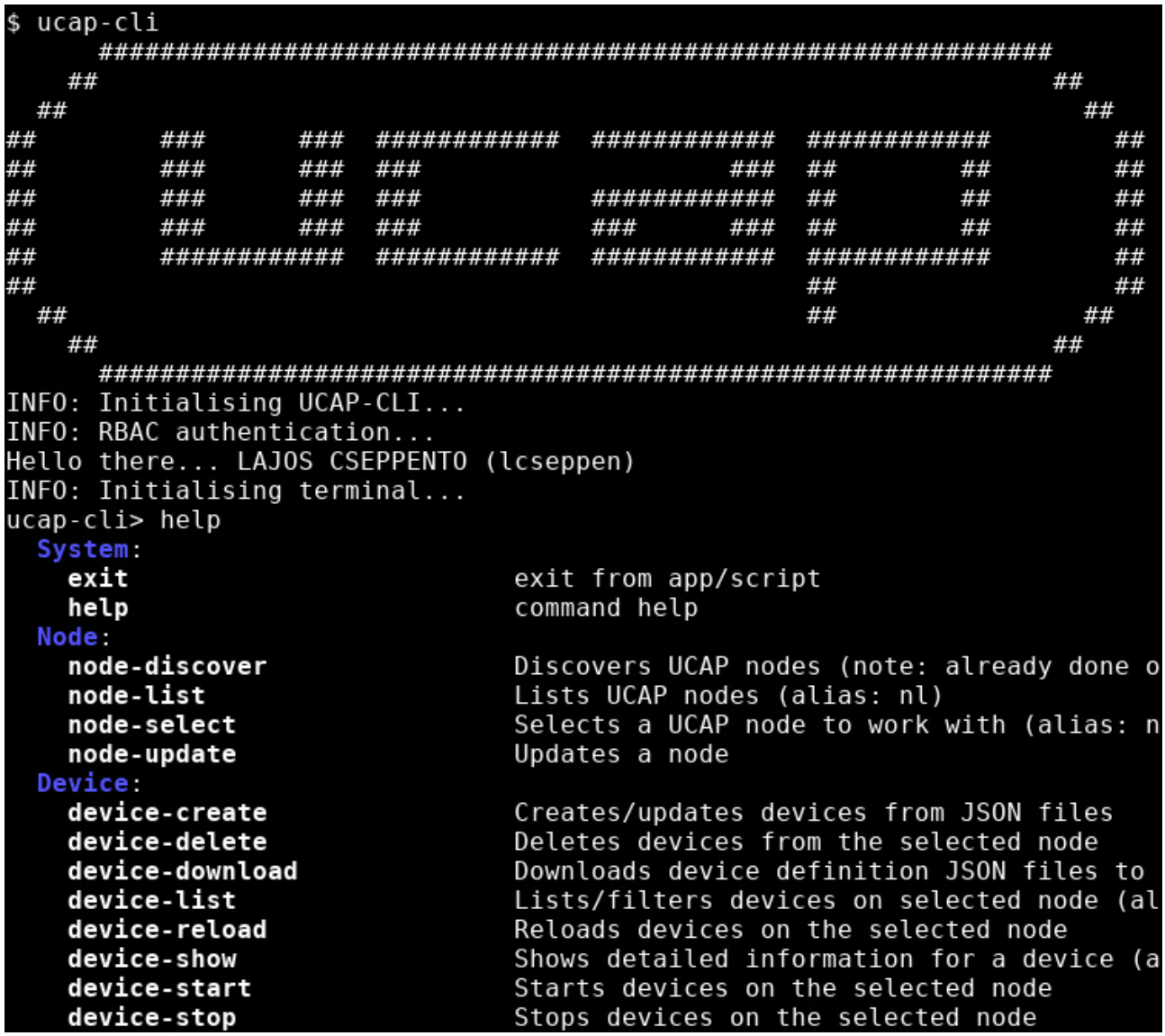


Figure 2: UCAP-CLI, the rich command-line tool.

DEVELOPMENT METHODOLOGY

An agile workflow was used to balance development and maintenance of UCAP with other ongoing projects. This comprised weekly planning meetings, daily stand-ups, frequent pair-programming, and code reviews using iterative development practices. A JIRA Scrum Board was helpful to keep focus and GitFlow practices to keep a high velocity. With limited resources and challenging deadlines, runtime verification favoured integration and system tests over unit tests, as experience showed that they were more efficient for discovering bugs and regressions during development.

TECHNOLOGICAL CHOICES

In 2018 the UCAP prototype was implemented with recent technologies (Java, Spring Boot, OpenFeign, Project Reactor and protobuf/gRPC). Several FaaS[7] solutions with Kubernetes were evaluated, however, at the end of 2018, Kubernetes-based solutions were considered as too risky to deliver within the strict deadlines, due to the significant involvement and long-term commitment required from multiple teams.

In early 2019, technological choices were finalised: Java, Spring Boot, OpenFeign, REST, JAPC and Controls Middleware (CMW). This eliminated partial duplication of communication features realised by gRPC in the prototype. Classic Java executor pools were favoured over reactive programming practices. The Open Services Gateway initiative (OSGi) framework [12] is used to support dynamic code loading (e.g., when users want to replace the transformation code on a running node). This resulted in simple and clear server-side code, while the OSGi layer is hidden from the user via a Gradle Plugin [13]. For UCAP-CLI we used the JLine library [14]. Java-Python communication is carried out over TCP and regarded as an internal protocol.

[7] Function-as-a-Service

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

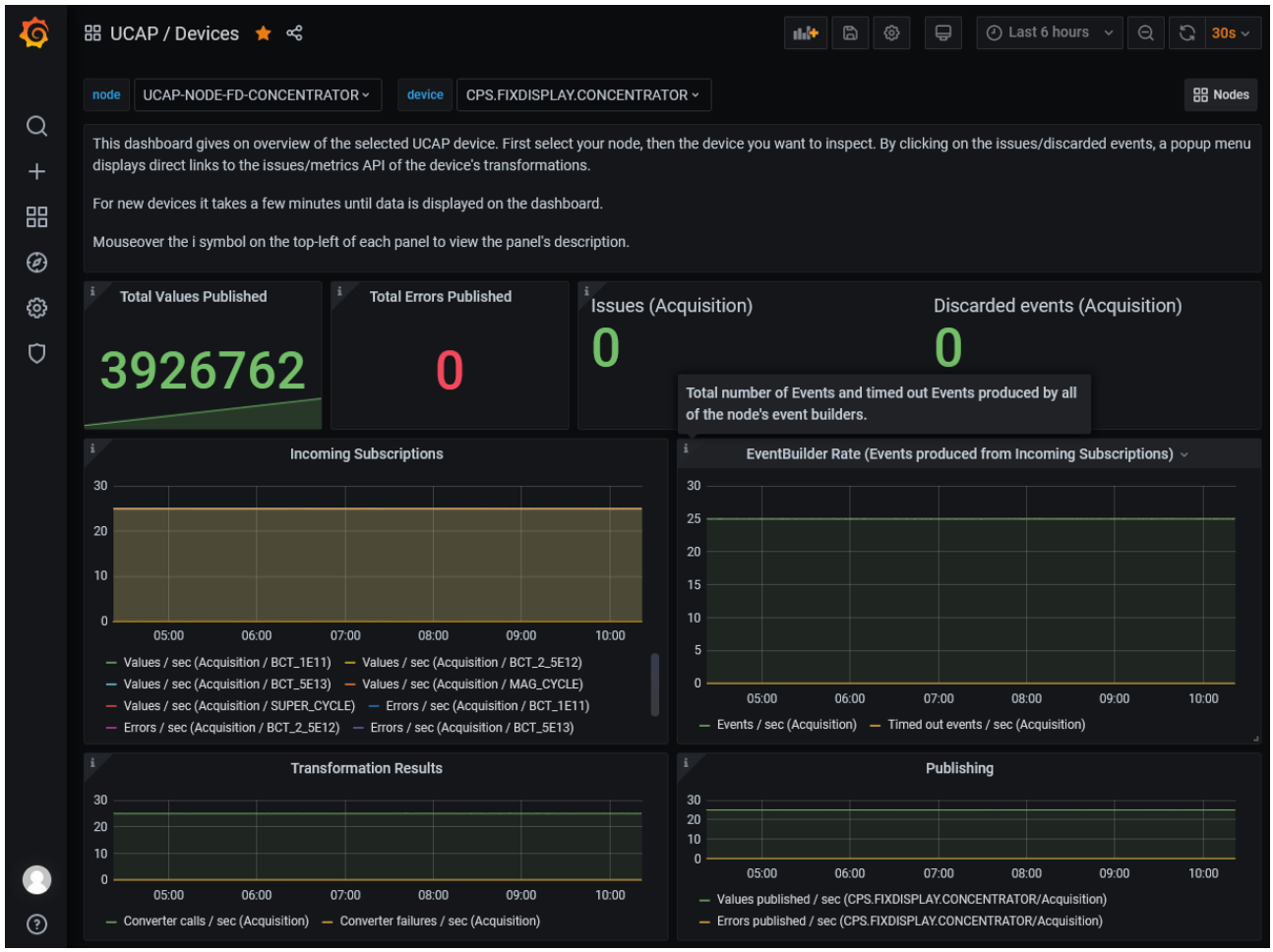


Figure 3: Grafana Dashboard of an Operational Device.

For code collaboration we leveraged GitLab. Highly automated testing is executed by a CI/CD pipeline on unit, integration and system test levels. Overall monitoring is based on the well-integrated Prometheus/Alertmanager/-Grafana [15] [11] stack, with JMX used for some expert interventions. Node maintenance is carried out based on UCAP-specific YAML files, while UCAP tooling takes care of tackling deployment and process management. This has proven to be a cost-effective solution.

The documentation is based on *Mkdocs Material* [16], adhering to the CERN design guide, encapsulating both Java and Python API documentations. As the last step of the development tasks and one of the least rewarding engineering tasks, focus was given to making documentation easy to write: the documentation source is hosted with the UCAP source code, thus documentation can be reviewed in the same Git merge request.

FUTURE WORK

In the first half of 2021, an extensive review of UCAP was conducted in order to reflect on the development and user experience gained so far and identify the next objectives. A tighter integration with the Controls Configuration Service [17] and support for real-time processing were important aspects identified from the review. From a UCAP service provider perspective, redundant middleware and availability of a computing cluster are key aspects that could lead to an even better UCAP service.

CONCLUSION

Used in Linac4 [18], PSB, PS, SPS and LHC – UCAP has been rapidly adopted across the CERN accelerator complex. The user community continues to grow, which is a clear indication of the success of the product. The high-quality standards followed during development have contributed to scarce bug reports and no service outage thus far. At the same time, UCAP has facilitated the removal of a significant number of custom and complex standalone systems, thus fulfilling the objective of providing a truly “Unified” Controls Acquisition and Processing platform.

ACKNOWLEDGEMENTS

The authors would like to express special acknowledgements to early adopters of the service. Their feedback contributed a lot to the product.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV039

REFERENCES

[1] V. Baggiolini, S. Jensen, K. Kostro, F. DiMaio, A. Risso, and N. Trofimov, "Remote Device Access in the New CERN Accelerator Controls Middleware", in *Proc. 8th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'01)*, San Jose, CA, USA, Nov. 2001, paper THAP003, pp. 496–498.

[2] V. Baggiolini, Lionel Mestre, "JAPC - the Java API for Parameter Control", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'05)*, Geneva, Switzerland, Oct. 2005, oral TH.1.5-8O

[3] L. Cseppentő, V. Baggiolini, E. Fejes, Zs. Kővari, and N. Stapley, "CBNG - The New Build Tool Used to Build Millions of Lines of Java Code at CERN", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 789–793. doi:10.18429/JACoW-ICALEPCS2017-TUPHA163

[4] P.Elson, I. Sinkarenko and C. Baldi, "Introducing Python as a Supported Language for Accelerator Controls at CERN", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV040, this conference.

[5] W. Sliwinski, K. Kaczkowski, and W. Zadlo, "Fault Tolerant, Scalable Middleware Services Based on Spring Boot, REST, H2 and Infinispan", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOBPP03.

[6] J. Lauener and W. Sliwinski, "How to Design & Implement a Modern Communication Middleware Based on ZeroMQ", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 45–51. doi:10.18429/JACoW-ICALEPCS2017-MOBPL05

[7] A. Schwinn *et al.*, "FESA3 – The New Front-End Software Framework at CERN and the FAIR Facility", in *Proc. 8th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'10)*, Saskatoon, Canada, Oct. 2010, paper WECOAA03, pp. 22–26.

[8] J. P. Wozniak and C. Roderick, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA163.

[9] Z. Zaharieva and M. Büttner, "CERN Alarms Data Management: State & Improvements", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper MOPKN011, pp. 110–113.

[10] P. Gajewski, S. R. Gysin, and K. Kostro, "Role-Based Authorization in Equipment Access at CERN", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper WPPB08, pp. 415–417.

[11] Grafana, https://grafana.com

[12] OSGi https://www.osgi.org/

[13] Gradle Build Tool https://gradle.org/

[14] JLine https://github.com/jline/jline3

[15] Prometheus https://prometheus.io/

[16] Material for MkDocs https://squidfunk.github.io

[17] L. Burdzanowski and C. Roderick, "The Renovation of the CERN Controls Configuration Service", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 103–106. doi:10.18429/JACoW-ICALEPCS2015-MOPGF006

[18] M. Hrabia, M. Peryt and R. Scrivens, "The Linac4 Source Autopilot", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper WEPV018, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

INTRODUCING PYTHON AS A SUPPORTED LANGUAGE FOR ACCELERATOR CONTROLS AT CERN

P. Elson, C. Baldi, I. Sinkarenko, CERN, Geneva, Switzerland

Abstract

In 2019, Python was adopted as an officially supported language for interacting with CERN's accelerator controls. In practice, this change of status was as much pragmatic as it was progressive - Python has been available as part of the underlying operating system for over a decade and unofficial Python interfaces to controls have existed since at least 2015. So one might ask: what really changed when Python adoption became official?

This paper will discuss what it takes to officially support Python in a controls environment and will focus on the cultural and technological shifts involved in running Python operationally. It will highlight some of the infrastructure that has been put in place at CERN to facilitate a stable and user-friendly Python platform, as well as some of the key decisions that have led to Python thriving in CERN's accelerator controls domain. Given its general nature, it is hoped that the approach presented in this paper can serve as a reference for other scientific organisations from a broad range of fields who are considering the adoption of Python in an operational context.

INTRODUCTION

The Python language, and specifically the cPython implementation for which this paper concerns itself, has one of the largest and fastest growing developer communities of any programming language [1]. It has been jokingly described as "the second best programming language for anything" [2] on the basis of its ability to adapt to a broad array of problems, including domains of particular interest to accelerator controls such as system automation, data analytics & machine learning, web services, and graphical user interfaces (GUIs). This flexibility comes less from the "batteries included" Python standard library, and is more a reflection of a rich suite of third-party packages available from the Python Package Index (PyPI).

CERN's Accelerator & Technology sector, referred to as simply "accelerator sector" in this paper, is responsible for the operation and exploitation of the whole accelerator complex, including the LHC, and for the development of new projects and technologies [3]. Python has been in use in the accelerator sector for a number of years, solving a diverse set of problems including fundamental physics simulations, gathering and analysis of data for machine development (MD) studies, rapid prototyping of services, GUI applications, and much more. For high-level operational accelerator controls at CERN, Java was the only supported programming language until the adoption of Python in 2019. The growth of Python has meant that more people are joining CERN with prior Python experience, and the simplicity of the language is compelling for the large user base of domain specialists for whom software development is a tool to achieve a specific task rather than being the main focus of their work.

Promotion of Python to "supported" status includes the provision of a software infrastructure to allow the operational 24/7 running of Python for mission-critical high-level accelerator controls, as well as infrastructure, such as tooling and a support service, to aid with effective development of such applications & services. Much inspiration was drawn from the existing Java service for accelerator controls, though to directly emulate the Java experience in Python would be to suffer the worst of both worlds and to risk missing the advantages of Python adoption. Therefore, a key objective was to preserve the spirit of Python in the service provided.

The objectives of this paper are to outline some common practices observed before the adoption of Python; to highlight some of the idiomatic Python practices that should be preserved; and at a high level, to present some of the infrastructure put in place to facilitate the use of Python in the accelerator sector at CERN. With this, it is hoped that other scientific organisations may be able to identify existing practices and potential areas of improvement in their own journey to bring Python into an operational context.

As a reflection of the computers in use for high-level controls in the accelerator sector, this paper predominantly focuses on x86-64 architecture, variants of Linux operating systems (OS) such as CentOS, and bare-metal deployments to machines which are often multi-purpose and multi-user.

A TYPICAL PYTHON STARTING POINT

Python Already in Use for Many Years

Python isn't new, and is almost certainly in use in most medium to large scientific organisations today. Typically installed by default in Linux-based operating systems, the barrier to first use of Python is relatively low – scripts can be executed directly by the Python interpreter for a rapid and iterative development experience. As they develop, re-use of functions and other definitions by gathering scripts together into a library of modules becomes desirable. Historically, this was achieved by setting the PYTHONPATH environment variable to point to a directory of packages for consideration in the Python import machinery. At the same time, it is common to want to make use of the rich set of 3rd party, open-source software (OSS) Python packages. On a machine dedicated to a particular task, it is likely that the OS package manager will suffice for the most popular libraries, but eventually

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

it is inevitable that one or more of the 300,000+ packages on the Python Package Index (PyPI) [4] is not available, at which point using the Package Installer for Python (pip) becomes essential.

A common solution to minimise the need for user installations is to provide a dedicated, feature complete, software stack, such as the Anaconda Python distribution [5]. Indeed, in the CERN accelerator sector a number of such distributions existed, yet the need to augment the environment with extra packages using pip remained.

Pip and Virtual Environments

Whilst pip is not part of the standard library, it is considered the de-facto package manager for Python and is the basis for a number of other popular Python tools such as tox, venv and poetry. This effectively makes pip an essential tool for everyday Python development.

The pip package provides a command-line tool from which other packages may be installed. The default file location used by pip when installing packages is the same location as where pip itself is installed. As a result, by default the installation of a package using pip requires user permissions of at least those of the user who installed pip in the first place. For Python & pip installed through the OS package manager, it is common, yet highly discouraged [6], to see the use of "sudo pip install" to grant pip sufficient privilege to install the packages into the system prefix. A common alternative solution is to request that pip installs packages into the "user site-packages" location found in the user's home directory. This location is typically included in Python's import search path, therefore both solutions result in packages being made globally available by default for all subsequent Python invocations. This means that the behaviour of an entirely independent Python application or service can be accidentally changed, without it itself having been modified. At CERN, since user home directories are on network mounted filesystems, the user site-packages location can impact applications running on different machines under the same user, and this in particular was found to be a common source of application instability.

The popular approach to avoid the global effect of pip installation is to use a virtual environment. Virtual environments can be created with the standard library "venv" package. A virtual environment is a prefix directory containing a link back to a "base" Python executable, and into which Python packages can be installed. In this way it is possible to have many virtual environments for a single base Python installation. When pip is installed in a virtual environment, packages can be installed without a global effect upon other Python invocations, and the environment is therefore considered "isolated". By design, it is possible for virtual environments to be created by users who do not have permission to write to the base prefix, thereby giving a clear path to allow a centralised base Python installation which can be extended by users through the use of virtual environments.

RUNNING PYTHON OPERATIONALLY

Advantages of Modules over Scripts

Whilst Python is an interpreted language, and Python script execution is undoubtedly the simplest way of running Python code, there are a number of advantages of running modules rather than executing scripts. When running a script:

- The directory containing the script is put on the Python path, with a higher precedence than any other directory, including that of the standard library. This can easily result in "name shadowing" and accidentally replacing parts of the standard library and third-party packages.
- There is no standard mechanism for sharing scripts (e.g. they can't be installed with pip), and no mechanism for declaring their dependencies.

A further observation is that scripts tend to encourage big blocks of linear code, with little factorisation and limited code reuse. For example, it is common to see a large directory of unmanaged scripts which each have subtle variations of one another. One desirable aspect of reuse, beyond the obvious ability to maintain functionality in a single place, is the ability to systematically test functionality through tools such as pytest. Whilst possible to write modules with exactly the same limitations as scripts, there is a much clearer consensus that Python modules should avoid the import-time behaviour in this way [7], and as a result, modules gathered into packages tend to be better equipped for effective code re-use.

As part of the guidelines developed for Python in the accelerator sector it was therefore recommended that the creation of packages, which are a collection of modules under a namespace, be favoured over the creation of collections of scripts. The immediate benefit to the user is that code can be easily re-used and tested, with declared dependencies being automatically installed alongside the code. Structurally, it enables relative imports in order to access sibling modules, and it avoids the need for later migration to a package structure as the code evolves.

The efficacy of this recommendation is hard to quantify in general, and strictly enforcing such a policy would be to the detriment of the flexibility that Python brings. Despite this, all known operational Python applications used in the CERN accelerator control room are deployed and run as packages in this way.

The Acc-Py Base Environment

A single, common, software stack containing all Python packages suffers with the problem that the software cannot be updated for one project without risking it breaking for another. When managing such a software stack, either the whole stack must be versioned such that newer iterations can be released without changing existing releases, or packages can be added but practically never removed

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

nor updated. From a developer perspective, the delay to have packages added to the stack inhibits productivity, and as a result, a mechanism to extend the stack locally is essential.

Since virtual environments have become a fundamental part of everyday Python development, they were the preferred approach in the accelerator sector in order to enable pip-based installation of both user-developed and third-party packages.

With the ability to extend a Python distribution through pip, choosing which packages should be in the base distribution is brought into focus. Providing a package in the base distribution has the advantage of guaranteeing that all applications use a consistent version and thereby increasing the chance that tools are able to interoperate. It was concluded that in general this advantage did not guarantee consistency in practice, and that the additional cost of maintaining an extensive base distribution was not justified. As a result, in the accelerator sector base distribution, also known as the Acc-Py base, the decision to include as few packages as possible was made. The packages provided today are limited to those which are hard to pip install in an effective manner, either because they need machine-specific optimisations or specific compilation options. There are currently just two such packages in the Acc-Py base distribution: numpy and PyQt5.

CERN's accelerator operating schedule, typically a 3-4 year physics run followed by a 2-3 year shutdown, requires relatively long periods of stability between upgrade periods. The lifetime of a particular Python version is defined at a maximum of 5 years [8], meaning that a Python version ready to be used from the end of a shutdown period is likely to reach end-of-life before the end of an operational run. This is a key consideration for exposed services, such as web applications, which must, as a result, be re-released on a frequent basis in order to pick up security patches of Python and third-party libraries. The majority of Python usage in CERN's accelerator sector, however, is related to data analysis and GUI applications. For these cases, the biggest impact of Python's end-of-life is that key third-party dependencies will gradually phase out their own support for ageing Python versions, for example Numpy Enhancement Proposal 29 [9] makes clear the intention to limit support within fundamental libraries such as numpy, matplotlib and scikit-learn for older versions of Python. This paints a clear picture that Python is a tool which can be useful for its ease and speed of development, but that there is an additional maintenance cost to be considered over the time frame of CERN's accelerator schedule to ensure that an application or service is adapted as new versions of Python and third-party libraries are released.

In order to facilitate this adaptation to newer Python versions, the Acc-Py base distribution has been designed to allow side-by-side installation with other Acc-Py base versions. Only at the end of an accelerator run will older versions be deprecated and removed, thereby giving application owners the freedom to choose the most appropriate time to adapt their code to a later version. Since no old versions are removed during a run, it can be guaranteed that a correctly isolated application which works at the start of an operational run will continue to work until the end of a run, but no such guarantees can be made for applications across multiple runs.

Whilst possible to install the base distribution on individual machines, the primary mechanism to deliver the distribution to all machines in the accelerator sector is via a shared network mounted disk. One major advantage of this approach is that deployed distributions are the same on all machines, and that updates are instantly available. When compared to a traditional local disk installation, downsides include that the network disk is a single point of failure, and that the added network latency could impact Python start-up time detrimentally. In practice, it has been found that having a single network location is a pragmatic approach and that the consistency across machines is particularly useful.

Tracing Python Startup

To provide an overview of how the various Python distributions are being used, Python start-up logging has been added to the Acc-Py base distributions. For each Python invocation, a centralised log entry is generated containing essential information including username and start-up time, as well as the packages installed in the environment. Currently up to 40,000 logs are received per day, providing a vital window into Python usage, supplying invaluable information when providing support, and highlighting areas of potential improvement to the underlying service.

Adding tracing to an existing service is generally seen unfavourably due to natural fears of an invasion of privacy. Doing so also introduces extra work in the Python startup, this fact also limits what can be reasonably traced. Therefore the general advice for those wishing to add Python startup tracing such as this, is to plan on providing the capability early in the adoption process.

An Internal Package Index and PyPI Proxy

When relying on pip for important work, it is considered good practice to run a PyPI proxy in order to ensure that any downtime of PyPI does not result in loss of productivity, or a loss in ability to deploy applications and services at key moments [10]. Furthermore, the computer network used for accelerator control at CERN prohibits internet access, providing extra motivation for having an internal PyPI proxy. As well as the ability to install public packages from PyPI, it is desirable to allow the installation of internal packages through pip. To solve both of these requirements, a Sonatype Nexus instance was installed. A package repository dedicated to internal releases and a PyPI proxy repository were created. These were then grouped together to form a single user-facing virtual repository.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

The fact that Python packages share a single global namespace is a cause of increasingly obscure and often comical package names on PyPI. Whilst the addition of a concept similar to Maven's groupId would help alleviate this, the algorithm used to group together two repositories into one is an important detail not to be overlooked from both a usability and security perspective [11]. In this specific detail, the Python Packaging Authority's devpi package is considered to be the reference implementation, against which Nexus has been found to fall short [12]. This is further exacerbated by the fact that pip itself is able to group together multiple repositories, but does so in an undefined and unexpected manner [13].

Once the package index is set up, pip can be pre-configured to use the package index with no need for further user intervention. For the Acc-Py base distribution and subsequently created virtual environments, configuration is done on a per-installation basis. One subtle and unexpected source of behaviour relating to this is that a suitably configured base distribution does not result in an equivalently configured virtual environment [14]. A patch of venv, the standard library package for virtual environment creation, in the Acc-Py base distribution was needed in order ensure that the configuration was copied into newly created virtual environments.

Along a similar line, the version of pip installed in the base distribution is not the determining factor for the version of pip in a newly created virtual environment. Instead, the version is encoded into the "ensurepip" standard library package, which is fixed at Python release time. Again, further patching of the Acc-Py base distribution was essential to ensure that modern versions of pip are being used in virtual environments.

Development and Deployment Tooling

In order to help users work efficiently and adopt best practice in Python development, a suite of development tools were created. This includes a tool to quickly set up a new Python package, with a well-defined project structure, and a placeholder for tests, sphinx-based documentation and GitLab-based continuous integration configuration.

In addition to the development tooling, a tool to deploy Python applications in a consistent and repeatable manner was developed. The tool, named acc-py-deploy, is comparable to pipx [15] in that applications are considered to be an extension of a Python package which can be installed into isolated virtual environments for later execution. Much like the poetry package [16], acc-py-deploy provides a mechanism to "lock" floating dependency versions into fixed versions. In the case of acc-py-deploy, this includes special behaviour to lock transient Java dependencies being used through JPype [17] such that an application may be deployed consistently as a virtual environment without the risk of dependencies differing between development and production. Deployed applications are run in full isolation mode, with both home-directory-based user site-packages and PYTHONPATH disabled. A unique feature of acc-py-deploy is its integrated ability to elevate to a dedicated service user for the purposes of deployment to a centralised network mounted location. In this way it is possible to have a common application deployment repository, with individuals able to manage the deployment of their own applications, but without needing to grant users direct write access to the installation directory. These features provide a strong guarantee that the application is deployed repeatably and consistently, and that the execution of an application is run in a way which avoids common user or environmental configuration issues.

The uptake of acc-py-deploy has been strong, with over 50 deployed applications in production in its first 12 months. The previously common issue relating to unpredictable application deployment and execution has been eradicated and has been replaced by a reliable and user-friendly operational experience.

PYTHON ADOPTION PRACTICALITIES

Fostering and Supporting a Community

The adoption and centralisation of Python for accelerator controls presents an opportunity to bring together a broad community with a common interest in the Python language and associated tooling. Regular community meetings are therefore arranged in order to provide a space to share news, knowledge and experiences relating to Python. Although the effort to coordinate contributions is relatively high, these meetings have been an invaluable medium for dissemination of key information, and they are an effective way to enhance Python skill across the accelerator sector.

The addition of a centralised support service is a fundamental part of officially adopting Python in the accelerator sector. Support includes resolving operational issues quickly and efficiently, as well as providing guidance to users developing software with Python. Prior experience has shown that providing first-rate and highly individualised guidance through a Python support channel can lead to requests reaching above-and-beyond the scope and capacity of a centralised support service. In an effort to foster a greater sense of community, and to mitigate against over individualising support, a community-centric chat space was created, into which questions and discussion is encouraged. Although a simple solution, the uptake has been excellent, and the sense of an active community comes across strongly. Indeed, it has often been the case that the community have been able to efficiently support one-another directly.

Centralisation of Pre-existing Functionality

When a community of users lack the tools needed to achieve a particular task, and there exists no service from which to request such tools be developed, it is common to see domain-specific solutions being created by users.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Whilst these tools are typically not designed for general purpose use, they provide invaluable insight into specific use cases.

In some cases, these solutions become popular amongst users and should be adopted for centralised support and maintenance. In the accelerator sector, one such tool, PyJapc, was developed by domain specialists in order to conveniently interact with the accelerator control system during machine development (MD) studies. The library is technically interesting as it directly bridges to the Java API for Parameter Control (JAPC) library, yet provides a less general API well-adapted for the use case at hand.

The centralisation of such tooling faces the risk of losing the strong user focus and potentially disenfranchising the original authors if the adoption is not done sensitively and pragmatically. In the case of PyJapc, whilst neither perfect nor general, priority was given to ensuring the existing behaviour was well-maintained, tested, and documented. Only after becoming extremely familiar with the code through bug-fixing and minor enhancements did larger API redesigns become a focus of effort. In turn this respect for what already exists buys credibility when proposing future changes, and it is hoped that this will ultimately lead to improved user uptake of future solutions.

Building-Out New Functionality Quickly

The breadth and depth of existing Java infrastructure for accelerator controls made Python adoption all the trickier, as replicating the infrastructure developed over decades of effort was neither practical nor efficient for the adoption of Python. Instead, emphasis was given to creating packages which bridge to existing implementations written in other languages.

For simple RESTful APIs with OpenAPI or Swagger definitions, openapi-generator was used to generate Python bindings. Some post-processing of generated code was added in order to improve code layout and to programmatically inject type annotations for an improved static analysis experience. Whilst runtime bindings would have been possible due to Python's highly dynamic runtime, it was found that the well-defined types from generated code result in a superior API, with support for convenient static analysis and auto-completion in an integrated development environment (IDE).

When exposing existing Java libraries, the binding technology chosen was the JPype package [17]. JPype integrates tightly with the Java Native Interface (JNI) to start a Java Virtual Machine (JVM) in the same process as the Python interpreter. As a result, data can be passed between Java and Python efficiently without the need to copy data. To improve the developer experience, a tool named stubgenj was created to generate Python type annotations for Java libraries. As a result, it is possible to run static analysis tools and IDE auto-completion when interacting with Java libraries through Python and JPype.

The other binding technology used to build-out capability quickly was PyBind11 [18]. The tool generates Python bindings to C++ libraries, based upon C++ declarations of the desired interface. These declarations can be relatively easily crafted to form intuitive and idiomatic Python APIs. The results of creating bindings through PyBind11 are compiled shared-object modules. In order to maximise compatibility of these modules and to allow installation on multiple Linux operating systems, the manylinux2014 [19] standard was used when publishing Python wheels [20].

CONCLUSION

The formal adoption of Python for accelerator controls at CERN was, on the surface, smooth and relatively seamless. Despite this outward appearance some significant culture shifts were introduced, including the removal of one-size-fits-all Python distributions in favour of virtual environments, the discouragement of script writing in favour of Python package creation, and the introduction of a community-focussed culture through user meetings and a collaborative chat space.

In the accelerator sector Acc-Py has been well-received, and as a result the use of Python is growing, users now have more flexibility to pick the best tools for their job, and most critically of all, the operational stability of Python applications and services has been ensured.

Python and its associated ecosystem are fundamentally open-source, and whilst costing more in the short-term, an effort has been made to report and fix bugs identified with the tools being used. In the long-term this strategy pays off, as it minimises the need for local patches and software bifurcation, and eases the future adoption of newer versions. A by-product of this is better software for all, including for other organisations following their own Python adoption story.

Some of the concepts presented around acc-py-deploy, particularly with regards to stability of execution and repeatability of deployment, can be reasonably compared to the benefits of containerisation. It is considered that the deploy concept from acc-py-deploy will work well as a build step for a container image, and that stable execution could then be handled by the container runtime. This is one particular area of potential future work for Acc-Py.

With key infrastructure in place, the Python adoption phase has now been replaced by a growth phase. Growth of the Python community in terms of number of users and the collective Python skill, and growth in terms of the quality & extent of the libraries provided to interact with CERN's accelerator control system.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV040

REFERENCES

[1] ZDNet review of Programming languages, https://www.zdnet.com/article/programming-languages-javascript-has-most-developers-but-rust-is-the-fastest-growing/

[2] D. Callahan, PyCon 2018 keynote, https://youtu.be/ITksU31c1WY?t=420

[3] Accelerator & Technology Sector, https://ats.web.cern.ch

[4] Python Package Index, https://pypi.org

[5] Anaconda distribution, https://www.anaconda.com/products/individual

[6] pip permissions overview, https://github.com/pypa/pip/issues/1668

[7] Imports should be "as free of side effects as possible", https://realpython.com/python-import

[8] Ł. Langa, PEP 602 - Annual Release Cycle for Python, https://www.python.org/dev/peps/pep-0602

[9] T. Caswell et al., Recommend Python and NumPy version support, https://numpy.org/neps/nep-0029-deprecation_policy.html

[10] H. Schlawack, Recommendation to run a private PyPI mirror, https://hynek.me/talks/python-deployments

[11] What is a dependency confusion attack?, https://secureteam.co.uk/news/what-is-a-dependency-confusion-attack

[12] Disclosure of dependency confusion vulnerability in Nexus for PyPI proxies, https://issues.sonatype.org/browse/NEXUS-24870

[13] Discussion of pip's index group strategy, https://discuss.python.org/t/dependency-notation-including-the-index-url/5659

[14] pip configuration not inherited in virtual environments, https://github.com/pypa/pip/issues/9752

[15] Install and Run Python Applications in Isolated Environments, https://github.com/pypa/pipx

[16] Python dependency management and packaging made easy, https://github.com/python-poetry/poetry

[17] JPype, a module to provide full access to Java from within Python, https://jpype.readthedocs.io

[18] Seamless operability between C++11 and Python, https://pybind11.readthedocs.io

[19] P. Moore, PEP 599 - The manylinux2014 Platform Tag, https://www.python.org/dev/peps/pep-0599

[20] N. Coghlan, PEP 427 - The Wheel Binary Package Format, https://www.python.org/dev/peps/pep-0427

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

MODERNISATION OF THE TOOLCHAIN AND CONTINUOUS INTEGRATION OF FRONT-END COMPUTER SOFTWARE AT CERN

P. Mantion†, S. Deghaye, L. Fiszer, F.Irannejad, J. Lauener, M. Voelkle
CERN, 1211 Geneva 23, Switzerland

Abstract

Building C++ software for low-level computers requires carefully tested frameworks and libraries. The major difficulties in building C++ software are to ensure that the artifacts are compatible with the target system's (OS, Application Binary Interface), and to ensure that transitive dependency libraries are compatible when linked together. Thus developers/maintainers must be provided with efficient tooling for friction-less workflows: standardisation of the project description and build, automatic CI, flexible development environment. The open-source community with services like Github and Gitlab have set high expectations with regards to developer user experience. This paper describes how we leveraged Conan and CMake to standardise the build of C++ projects, avoid the "dependency hell" and provide an easy way to distribute C++ packages. A CI system orchestrated by Jenkins and based on automatic job definition and in-source, versioned, configuration has been implemented. The developer experience is further enhanced by wrapping the common flows (compile, test, release) into a command line tool, which also helps transitioning from the legacy build system (legacy makefiles, SVN).

INTRODUCTION

Front-End Computer Software Development at CERN

CERN's Front-End Computers (FECs) are disk-less computer crates which host electronic cards connected on a back-plane. The software running on these computers typically uses a framework such as Front-End Software Architecture (FESA) [1,2] to interface with:

- The upper layer of the control system (settings management, timing, network (Remote Device Access (RDA3) protocol [3]), logging, post-mortem, machine protection, etc.)
- The cards' driver (C library) to drive the equipment.
- The OS (Linux, CentOS 7 with RT kernel) and framework (FESA) are designed to provide near real-time execution of the tasks through scheduling, thread priorities, and optimisation: this is a strong reason (amongst others), for using a performance-oriented language like C++ to build the software.

The production FECs run on CentOS 7, so the software must be built for that target, ensuring compatibility with the system's libraries (especially libc) and ABI (changes in ABI for *C++11 support*).

Software built using the FESA framework consists of an executable that is statically linked against the framework's libraries (versioned headers and *.a* archive files). The framework libraries themselves depend on a collection of middleware libraries provided by different teams across different groups.

The FESA framework is mostly used by equipment developers who are not full-time software engineers. As such, the framework providers aim to offer tooling that promotes best practices (e.g. source code versioning, releasing, tagging) and minimises human errors.

NEED FOR MODERNISATION

After almost two decades of building C++ software with Makefiles, a well-deserved modernisation was needed. A Continuous-Integration (CI) solution, based on a shared central Bamboo Server instance, was put in place almost ten years ago. A general move away from Bamboo to Jenkins or Gitlab CI for Controls software also needed to be taken into consideration. Additional objectives were to ensure that the new solution provides a better dependency management and ensure a smooth transition for our users.

Dependency and Toolchain Management

The correct execution of FEC software requires the binaries to be built using consistent versions of the dependencies. At the lowest level, this means that versions of the dependent libraries should be both binary and functionally compatible. However, a complex dependency graph means it is not easy to ensure, especially if no compilation/linking errors are raised at build time. Dependency management entails two aspects:

1. Knowing where to find/store artifacts (header / library files) from the build system.
2. Being able to check the consistency of dependencies and versions in the dependency graph.

Beyond ensuring production software is built correctly, strong dependency management is very useful to the developer. When working with local copies of a sub-set of a dependency graph, developers are one step away from so-called "dependency hell". Without automatic dependency management developers need to ensure that all libraries used locally are compatible, which entails editing makefiles and building/rebuilding lots of dependencies before having a working setup. Compilation time in such cases is not negligible. The process is also error prone, often requiring to re-build several dependencies after each correction. Figure 1 highlights the difficulty of modifying dependencies by hand by showing the complexity of the dependency graph for a representative example of FEC software.

† pierre.mantion@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

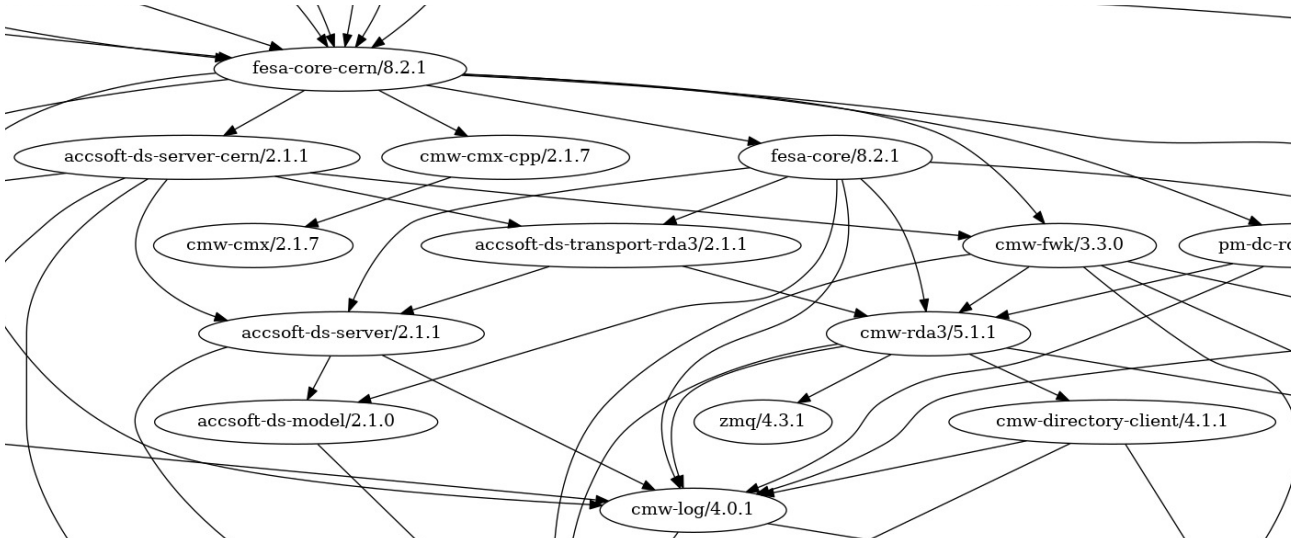


Figure 1: example of dependency graph for a FEC software (cropped)

Binary compatibility of the dependency libraries requires building all dependencies with compatible toolchains (compiler tools and settings, including libc and *C++*11 ABI [4]). The archive file (.a) artifacts of dependencies can be used by the software that is including them without many checks regarding compatibility.

New *C++* standards require more recent compilers, but the ABI must still be compatible with the runtime environment. The CentOS7 toolchain (gcc 4.8, "old" ABI [4]) is the current standard at CERN, but production software can also be released using a more recent compiler (gcc 7). Since CentOS 7 was released several years ago (2014) a transition to a new supported toolchain must be anticipated to ease equipment groups migration efforts and ensure smooth operations.

The rest of this paper will describe how Conan [5] and CMake [6] can be leveraged to meet the aforementioned challenges and objectives, and improve developer experience.

Introduction of CMake as a Modern Build Tool

While Make is still a standard way of building software, it is a fairly low-level tool. CMake brings deeper domain knowledge of C/C++ projects, is extremely popular and very well supported by IDEs, and has a permissive BSD license.

"CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice." – https://cmake.org

As a standard tool with good public documentation, CMake is favourable over the legacy in-house makefile hierarchy for a multitude of reasons:

- Better public documentation means less user support for FESA framework providers.
- Better integration with modern and powerful IDEs such as CLion increases developer productivity and satisfaction.
- The experience gained working with CMake is more valuable for trainees and developers in terms of transferable skills beyond CERN.

A Command-Line Helper

Although aiming to increase the ease and quality of development, transitioning to new tools like CMake and Conan represents a challenge, especially for part-time developers, where adaptation and practice are required to master the new tools. To minimise this challenge a command line (CLI) tool, named Codeine is provided to simplify and standardise workflows e.g:

- Create a new project with a standard file layout.
- Wrap complex commands.
- Provide safe defaults (CentOS 7 compiler toolchain).
- enforce a consistent release flow (versioning, build, tag VCS, release location, structure of released artifacts).

As such, Codeine can help the transition to a new compiler toolchain by providing setting profiles and migration helpers.

Renovation of the CI / Testbed

Tooling that can ensure a correct and reproducible build by design does nothing to help avoid functional failures. Testing the software is critical and making the test feedback loop as short as possible increases development efficiency. Having convenient tooling to setup and run tests helps ensure that tests are written, ran and results checked. Continuous Integration (CI) has been a standard practice in software development for a long time, and supporting tooling for it is evolving.

CERN's front-end software is well tested by an extensive collection of tests built over years, automatically executed from an ageing Bamboo platform [7]. Besides obsolescence of this platform (licence renewal, end of life support from Atlassian [8]), new

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

paradigms such as storing test job configuration together with the versioned source code have appeared and make developers' lives easier, by automating the configuration of the test plans / jobs. This approach saves thousands of clicks in the Bamboo interface to configure the dozens of CI jobs.

MODERNIZATION OF THE DEVELOPMENT ENVIRONMENT

Legacy Environment

The FEC software development environment at CERN (Figure 2) is tightly coupled to a Network File System (NFS) and to Virtual Machines (VPCs) that are setup for FEC development. The typical developer connects to their VPC and runs a dedicated distribution of the Eclipse IDE to develop. The NFS is mounted automatically on the developer VPCs. The development tools and dependencies (IDE, makefiles, dependent libraries) are either hosted on NFS, or depend on data or binaries hosted there.

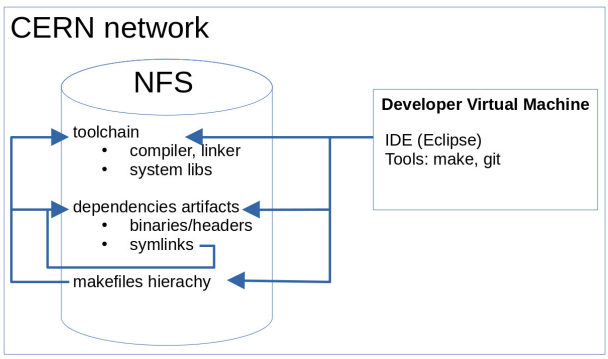


Figure 2: High-level view of the legacy development environment

The in-house makefiles provide build and release functionality, which is possible by knowing the layout of the dependency artifacts and the toolchain location on NFS. It knows the base path for artifacts (headers/libs) at build time, as well as where to store an artifact upon release. There is no enforcement of dependency consistency, nor of toolchain/compiler settings. As it predates Git, it only supports SVN.

This setup makes it easy to provide a functional development environment to the developer, but at the cost of flexibility:

- The development machine requires access to NFS, which is accessible only from the CERN network.
- Connecting to the development VPC is inconvenient (VNC, ssh -X), especially when the network access is with reduced bandwidth (e.g. teleworking), or when using multiple screens (VNC limitations).
- The custom layout of the dependencies folder on NFS makes it incompatible with most standard tools (IDEs, build systems) without extra integration work and project metadata duplication.
- The need to access NFS makes offline development or CI on the cloud either impossible or very inefficient.
- The Eclipse IDE for C++ is falling behind other C++ IDEs in terms of features and ergonomics (e.g. CLion, QtCreator, VSCode)

Modernizing the developer's experience entails keeping the ease of access of the development environment, while ensuring the continuity and quality of production artifacts. Regarding the development tools, that means providing more flexibility (e.g. offline work, local toolchain, choice of modern IDE, etc.). On the CI side, it means an easy initial setup and no redundant configuration tasks, as well as support for modern development workflows based on branches and merge requests.

Modernization of the Development Tools

Replacing the makefile hierarchy with CMake brings standardisation to what was an in-house custom project structure. Moreover most C++ IDEs have good support for CMake projects, thus enabling much better IDE support out of the box.

Alongside CMake, Conan is used to alleviate other pain points of the legacy setup.

"Conan is a MIT-licensed, Open Source package manager for C and C++ development, allowing development teams to easily and efficiently manage their packages and dependencies across platforms and build systems." - https://conan.io

- Conan describes package dependencies precisely in its recipe file (a text or python file called *'conanfile.[txt/py]'*), and is able to check the consistency of the versions in the dependency graph, thus avoiding the "dependency hell".
- Conan enforces strong but configurable versioning of binary packages. The binary package's ID includes a hash of compiler settings to ensure binary compatibility of the produced artifacts
- Through the given metadata, Conan provides convenient ways to either fail a build if a dependency cannot be satisfied (wrong version, wrong toolchain/compiler settings), or build the dependency from source
- Conan can package binary artifacts and share them through Conan repositories. Binary packages can be downloaded and build artifacts uploaded via HTTP(S) with appropriate authentication and authorization. Conan repositories are implemented by various products, including Artifactory, Gitlab, Conan-server, Nexus. With no need for NFS access anymore, cloud and offline builds are much easier.
- Setting up a local build environment for a project with arbitrary dependencies is as simple as running "conan install"

Renovation of the CI and Testbed

The decision to move away from Bamboo was triggered by the need to migrate to a newer version, and to

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

renew the licenses. Migration would have been a significant effort that already justified considering other platforms. Two options were evaluated, both already used at CERN and both without extra license fees: Jenkins (open-source) and Gitlab CI (deployed globally by the CERN IT department).

The legacy version of Bamboo in use so far required many slow interactions with the user interface to configure jobs and it had limited support for merge requests and branches. Therefore, the renovation was also an occasion to benefit from features brought by more modern tools. Below is a list of features expected from a CI platform:

- In-code configuration (.gitlab-ci.yml / JenkinsFile) and auto-configuration (zero-click) of the CI jobs
- Support for complex pipelines (serial and parallel stages, optional stages)
- Automatic support for branches and merge requests
- Support for triggers (an upstream jobs triggers the build of a downstream dependent job)
- Support for custom agents/executors
- Access rights management, integration with CERN's SSO
- Support for branch heavy workflows (e.g. libA on branch b1 will automatically pick up dependencies on the b1 branch if it exists)

IMPLEMENTATION STORIES, LESSONS LEARNT

The Bamboo to Jenkins Story

In terms of features, both Gitlab CI and Jenkins offer all basic features of a CI platform. What tipped the balance in favour of Jenkins was the fact that it is open-source, and not dependent on licenses from Gitlab. By using an open-source product we limit the risk of having features moved behind a higher-priced licence tier in the future. However, deploying dedicated Jenkins instances means additional work to build and maintain them versus the IT-managed Gitlab CI platform. Gitlab integrates vertically a lot of features that could be useful but which increase vendor lock-in (e.g. docker/Conan registries). The documentation of Gitlab is more complete and much better structured than that of Jenkins, especially regarding the domain specific language (DSL) to define pipelines. The Jenkins ecosystem is made of a lot of plugins. Support, documentation and compatibility of plugins can be a challenge. In practice, these are not blocking points, since most configuration is done once, and experience is shared within the team. Additionally it was decided to keep most of the CI jobs as simple shell command invocations, which is very portable across CI systems in case it is needed to migrate again in the future. This reduces vendor lock-in, and facilitates running jobs manually for testing or debugging.

The Jenkins master is deployed as an OpenShift pod, whose configuration is completely described as code. Deployment on OpenShift means no need to deploy/maintain hardware. Configuration as code mitigates the lack of durability of pods on OpenShift, as well as making it easy to reuse (e.g. for another project/team) and easy to bring back up in case of major failure. The main challenge here was to find how to serialize the configuration (Configuration as Code) of Jenkins and its plugins, because the format and commands to use are barely documented, requiring long searches for help on the internet/stackoverflow [9], as well as a lot of trial and error.

Beyond the basic Jenkins installation, the most important plugin is the Gitlab Branch Source plugin [10] that allows to simply point to a Gitlab folder from Jenkins, and auto-configures the CI jobs for all repositories, branches and merge requests in that Gitlab folder.

Codeine: CLI helper

Codeine (a play on words between "code" and the painkiller drug) is an idea to make development painless, especially considering two types of software developers:

- Framework and tooling providers
- Equipment developers

The distinction is often blurred depending if the focus of the team is on software engineering or more direct operational commitments. The goal with Codeine is to make the former's life easier, while letting the latter benefit from technical improvements with no/minimal impact to their work. Considering the ageing makefile-based build framework, the original idea behind Codeine was to wrap and abstract the steps of software development in the tool (create empty project, build, test, release) to be able to evolve the underlying implementation bricks with no impact on developer's workflow and habits. Codeine is based on a file describing the project, called *'product.xml'* which, in a first approach, contains metadata about the project: name, version, VCS repository, type (library or executable).

The first Codeine implementation re-used the custom makefiles to perform operations on the project (e.g. build, release). This approach helped consolidate the knowledge about existing workflows: inputs, outputs, dependencies. It also facilitated the migration of repositories from SVN to Git, based on the metadata of the *'product.xml'*. By implementing Git repository support in Codeine, the risky and complicated task of adding sub-par Git support to the legacy makefiles was avoided. After working on implementing the artifact release flow in Codeine, further metadata was added to the *'product.xml'* to structure the released artifacts (e.g. does the artifact require extra files beyond the binary to be delivered? symbolic links defined?). Metadata identifying the dependencies of the product were also added to *'product.xml'* in anticipation of better and/or automated dependency management. In summary, this version of Codeine fully supported the legacy makefiles framework, additional support for Git repositories, and a structured release process.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A First Proof-of-Concept

Building on the Codeine base described above, a proof of concept was launched to integrate modern tools into the development workflow and validate the feasibility of using Conan to manage dependencies and CMake as a more modern build system.

From the outset Codeine was intended to support the transition between the makefile-based build system to a more modern one. It was anticipated that the existing *'product.xml'* content should be sufficient, or require only minor adaptation, however it was soon discovered that this was not the case. The first *'product.xml'* change required was to include an element to tell Codeine which build system it should use. This was considered a necessary hole in the abstraction layer of Codeine.

Having decided on trying out Conan to manage dependencies, the proof of concept used Conan through Codeine to build most of the framework development team's libraries. The first step was learning about Conan usage, while the second milestone was letting Conan know about Codeine's metadata, by using Conan's mechanism to extend python conanfiles.

Since Conan is not a build system for C++, but rather generates configuration for one, and CMake, among other advantages, has good support from Conan, it was decided to use CMake from Conan to build libraries. This makes for a complex flow: Codeine calls Conan, which uses a Codeine extension (python) to understand the *'product.xml'*, and generate files for CMake, which in-turn is used to build the library.

Although complex, this proof of concept already brought several advantages:

- Strong dependency and toolchain management thanks to Conan.
- Standard dependency declaration
- Standard artifact packaging, ready to be consumed by a downstream project (using Conan).
- Conan's remote features to upload/fetch packages from an artifact repository, in turn enabling jobs isolated from the legacy NFS repository to fetch artifacts through HTTP.

However this proof of concept also showed that abstracting over both a loose legacy system and a stricter modern one is very hard. Several complications to Codeine were implemented to support both systems, in an imperfect way. Furthermore, the generated CMake files were complex and did not provide the extensive IDE support expected (e.g. QtCreator supports CMake based projects, but was not able to parse the generated CMake files satisfactorily)

Lessons learnt:

- A floating period transitioning back and forth between the legacy and new system is to be avoided
- Good abstractions are hard, especially over vast and/or vague domains such as "developer experience"
- Retaining legacy comptibility takes a lot of effort, with little reward
- Conan brings a lot of structure to a project with its strict dependency management, however, this does not fit nicely with the loose structure of the legacy makefiles, which harmed the goal of Codeine being a perfect abstraction over the build system.
- More abstraction layers make debugging harder (why/where did a build fail: in Codeine? in CMake? in Conan?)

Looking Forward: Second Proof-of-Concept

Given the experience of the Conan + Codeine + CMake proof-of-concept, a next iteration is foreseen, with aims to reduce complexity, and avoid the leaky abstraction problem posed by Codeine's *'product.xml'*.

Instead of Codeine being a complete wrapper over the development workflow, it will be re-focused on the parts missing from other tools: creation of new empty projects, and managing the release process (e.g. ensuring code is committed, pushed, that the code repository is tagged with the release version, etc.). This reduced scope of Codeine will allow to simplify the usage of tools like Conan and CMake, making the most of them without being limited by the incomplete abstraction that Codeine's *'product.xml'* tried to be. The advantage of using publicly available and documented tools instead of Codeine-as-a-wrapper are:

- More flexibility to the developer.
- Less support for the providers of Codeine.
- More publicly available support for the tools (e.g. Conan's and CMake's documentation, experts on StackOverflow [9]).

CONCLUSION

Software engineering best practices evolve and improve with time, as do automation and tooling to support developers. Embracing such evolution in CERN's FEC software development environment brings added value, but given the variety of constraints,the modernisation is both technically and organisationally challenging. Forced migrations such as the CI/Testbed are a good opportunity for improvements. Modernisation is a continuous iterative process, and while changes are disruptive and require stakeholder buy-in, a flexible architecture allows for iterative changes to be rolled-out progressively.

Beyond developer productivity and satisfaction, having a modern development environment is also more attractive to potential candidates, for whom experience with industry standard tools is more valuable.

REFERENCES

[1] M. Arruat *et al.*, "Front-end software architecture", in *Proc. ICALEPCS'07*, Knoxville, Tennessee, USA, Oct. 2007, paper WOPA04, pp. 310-312. https://jacow.org/ica07/PAPERS/WOPA04.PDF

[2] A. Guerrero *et al.*, "CERN front-end software architecture for accelerator controls", in *Proc.ICALEPCS'03*, Gyeongju, Korea, Oct. 2003, paper WE612, pp. 342-344. https://jacow.org/ica03/PAPERS/WE612.PDF

[3] J. Lauener and W. Sliwinski, "How to design & implement a modern communication middleware based on ZeroMQ", in

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV041

Proc. ICALEPCS'17, Barcelona, Spain, Oct. 2017, pp. 45–51. doi:10.18429/JACoW-ICALEPCS2017-MOBPL05

[4] GCC Dual ABI, https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html

[5] Conan Package Manager, https://conan.io

[6] CMake, https://cmake.org

[7] Bamboo, https://www.atlassian.com/software/bamboo

[8] Attlassian Server end of support, https://www.atlassian.com/migration/journey-to-cloud

[9] Stack Overflow, https://stackoverflow.com/

[10] Gitlab Branch Source plugin, https://plugins.jenkins.io/gitlab-branchsource

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PLCverif: STATUS OF A FORMAL VERIFICATION TOOL FOR PROGRAMMABLE LOGIC CONTROLLER

I. D. Lopez-Miguel*, J-C. Tournier, B. Fernandez, CERN, Geneva, Switzerland

Abstract

Programmable Logic Controllers (PLC) are widely used for industrial automation including safety systems at CERN. The incorrect behaviour of the PLC control system logic can cause significant financial losses by damage of property or the environment or even injuries in some cases. Therefore ensuring their correct behaviour is essential. While testing has been for many years the traditional way of validating the PLC control system logic, CERN developed a model checking platform to go one step further and formally verify PLC logic. This platform, called PLCverif, was first released internally for CERN usage in 2019, is now available to anyone since September 2020 via an open source licence. In this paper, we will first give an overview of the PLCverif platform capabilities before focusing on the improvements done since 2019 such as the larger support coverage of the Siemens PLC programming languages, the better support of the C Bounded Model Checker backend (CBMC) and the process of releasing PLCverif as an open-source software.

INTRODUCTION

Programmable Logic Controllers (PLC) are widely used for industrial automation including safety systems at CERN. The incorrect behaviour of the PLC control system logic can cause significant financial losses by damage of property or the environment or even injuries in some cases. Therefore ensuring their correct behaviour is essential. While testing has been for many years the traditional way of validating the PLC control system logic, it is often not sufficient as the sole verification method: testing, even when automated, can not be exhaustive, thus can not guarantee the correctness of a logic. Some types of requirements, such as safety (i.e. an unsafe state can never be reached) or invariant (formulas which shall be true over all possible system runs), can be very difficult, if not impossible, to test. Model checking is a formal verification technique which complements the testing activities in order to fully validate and verify a PLC control system logic. Model checking assesses the satisfaction of a formalised requirement on a mathematical model of the system under analysis. It checks the requirement's satisfaction with every input combination, with every possible execution trace. In addition, if a violation is found, a trace leading to the violated requirement is provided. The main hurdle to the widespread usage of model checking within the PLC community is twofold: (1) the mathematical model representing the system under analysis can be difficult to write and requires in-depth understanding of the model checking tools; and (2) many real-life PLC logics are too complex and face the state-space explosion problem, i.e. the number of possible input combinations and execution traces is too big to be exhaustively explored.

In 2019 CERN developed the PLCverif platform with the goals of easing the usage of model checking tools for the PLC developers community by automating the translation of the PLC programs to their mathematical models and to implement several abstraction algorithms to limit the state-space explosion problem. Since September 2020, the platform has been released under an open source license to foster the usage and the development of the tool within the PLC community. The objective of this paper is to give a status of the PLCverif platform focusing on the latest developments improving the usability and performance of the tool.

The rest of the paper is organized as follows: Section *PLCverif Overview* gives an overview of PLCverif to better understand the scope and the architecture of the platform. Section *Open Source Release* focuses on the open source release of PLCverif by describing the process of releasing the source code and presenting the code organization. Finally sections *Latest developments* and *On-going Challenges and Developments* present respectively the latest and ongoing developments.

PLCverif OVERVIEW

This section gives an overview of the PLCverif platform [1] before presenting the latest developments.

Verification Workflow

Out of the box, PLCverif offers a model checking workflow for the analysis of PLC programs. The verification workflow is shown in Fig. 1 and it has the following main steps:

1. **PLC program parsing.** PLCverif parses the PLC program (located in one or several files) to be analysed. By choosing the entry point of the verification, the analysis can be limited to a part of the program. The parsed PLC program is automatically translated into a mathematical, control flow-based representation, producing so-called Control Flow Automata (CFA). This precise description will serve as the basis for the analysis.

2. **Requirement representation.** The user should describe the precise requirement to be checked. This, however, does not mean that the user needs to describe the requirement using mathematical formulae. Currently, two requirement description methods are supported:

   - Assertion-based requirements: special comments in the source code (e.g. `//#ASSERT On<>Off`)

* ignacio.david.lopez.miguel@cern.ch

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV042

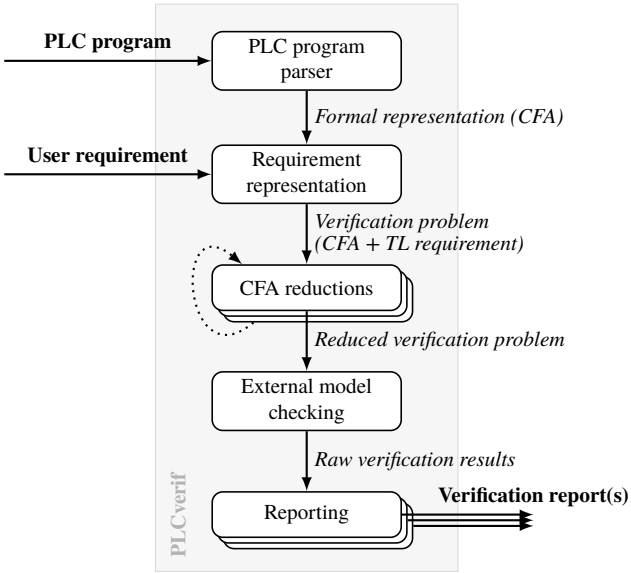


Figure 1: Formal verification workflow of PLCverif

can describe expressions (invariants) which are expected to be always satisfied at a given location of the program. The verification job will then check if the violation of any of the selected assertions is possible.

- Pattern-based requirements: the user chooses a requirement pattern that is a precisely-phrased plain text sentence with some placeholders, e.g. "*If α is true at the end of the PLC cycle, then β should always be true at the end of the same cycle.*". The gaps in the requirement pattern (α and β in the previous example) shall be filled with expressions over the PLC variables. For each requirement pattern, a defined temporal logic representation is defined which will be used in the next steps.

If needed, new types of requirement representations can be defined, adapted to the specific needs.

3. **CFA reductions.** The formal, precise CFA representation of the program, including also the requirement, may need to be reduced in order to make the verification feasible and efficient. These reductions will not change the verification result for the given requirement; however, they may remove parts of the program which do not influence the result of the currently checked requirement [2].

4. **External model checking.** The model checking itself is performed by widely used model checker tools. In this step, *(i)* the reduced CFA is translated into the input syntax of the chosen model checker tool, *(ii)* the model checker tool is executed, and *(iii)* its output, notably the counterexample if available, is parsed to PLCverif's internal representation.

   Currently the following external model checkers are supported: NuSMV [3], nuXmv [4], Theta [5] and CBMC [6]. These model checkers have different strengths and weaknesses. In addition, not every feature is supported by every model checker.

5. **Reporting.** The last step of the formal verification workflow is to produce verification reports. Some of these reports are in human-readable form and target the user of PLCverif. Other reports are machine-readable and serve as descriptions for the execution environment or as artifacts for later summary reports.

OPEN SOURCE RELEASE

PLCverif is available publicly since September 2020 along with its source code under an EPL-2.0[1] license on GitLab[2]. This section presents the reasons for open sourcing PLCverif, as well as the process to choose the open source license and the code organization as found in the GitLab repository.

Motivations

PLCverif has been entirely developed at CERN and therefore fits CERN's needs. Nevertheless, the platform could be beneficial to two different communities outside CERN: the PLC developers community and the model checker community. For PLC developers, PLCverif could be used out of the box if the language used is among the ones already supported, i.e. Siemens Statement List (STL) and Siemens Structure Control Language (SCL). The current coverage[3] of Siemens STL and SCL in PLCverif is currently at 66% and 40%, respectively, for S7-300/400 PLCs, and at 55% and 25%, respectively, for S7-1200/1500 PLCs. This covers most of the functionalities of PLC programs developed at CERN as the instructions implemented represent the core of those languages. However, support for a missing instruction could be easily added if needed. Similarly, the support of a new programming language could also be developed taking as a reference the current implementation for the Siemens languages. For the model checking community, PLCverif offers the possibility to be integrated in a platform verifying real-life PLC code: it is a great opportunity for this community to test new algorithms or improvements of existing model checkers.

License Selection

From the different open source licenses available, the choice has been made to release the PLCverif platform source code under the Eclipse Public License 2.0 (EPL[4]). This license is similar to the GNU General Public License (GPL[5]) but allows to link the code to proprietary applications: it then allows to use and extend the tool, even for

[1] https://www.eclipse.org/legal/epl-2.0/
[2] https://gitlab.com/plcverif-oss
[3] Coverage was calculated as the number of instructions that are implemented in PLCverif vs. the number of instructions in a given grammar for a given PLC. Binary arithmetic operations like addition or subtraction, and binary logic operations like conjuntion or disjuntion are not taken into account when the coverage for SCL is calculated. That is the main reason why the figures for SCL are much lower than for STL.
[4] See footnote 1.
[5] https://www.gnu.org/licenses/gpl-3.0.en.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

commercial purposes. The reason for choosing the EPL-2.0 license was driven by the fact that is a weak copyleft license, but also that most of the components used by PLCverif are already under the EPL-2.0 (more details in [1]).

GitLab Repository

The PLCverif platform has been developed within the Eclipse ecosystem: it is mainly written in Java (Java 11), Xtend and Xtext. The modularity is assured via the split of the code among several distinct projects and Eclipse plugins. The main projects of the platform are:

- PLCverif Backend[6] is the core of the PLCverif platform and is responsible for all the CFA manipulations represented in Fig. 1. It is also responsible for interacting with the external model checkers via dedicated Eclipse plugins.
- Siemens Support[7] is responsible for the parsing of the Siemens STL and SCL code, i.e. transforms the PLC user code into the PLCverif internal CFA representation.
- PLCverif Frontends[8,9] are the visible parts from a user point of view. The GUI project provides a graphical application embedding a PLC code editor, a specification requirement editor and a report visualization part. The CLI project is the way to execute the PLCverif workflow via the command line allowing the execution of headless verification jobs such as in a CI/CD (Continuous Integration / Continuous Deployment) pipeline for PLC code.

LATEST DEVELOPMENTS

Since the publication of [1] in 2019, several improvements have been made. Some of them are summarized below:

- The C code used to run CBMC was originally produced directly converting the intermediate model to a C code using *goto* instructions. CBMC is however not efficient with this kind of programs since it is not able to find loops. This method has been changed in order to produce a structured C code without *gotos*, being able to efficiently use the option *–partial-loops* of CBMC. With this option, CBMC will execute loops only partially. The disadvantage of this option is that the counterexample might be spurious.
- In order to confirm the feasibility of a counterexample produced by PLCverif when a property is violated, it is common to try to reproduce that situation in a real PLC or via simulation. In order to automate this process, it is now possible to automatically generate a file with the values of all the variables that can be used as an input to the simulator or to the real PLC.
- Safety programs in Siemens are written in function block diagram (FBD) language. After exporting them with *OpennessScripter*[10], an XML file is produced. A new feature in PLCverif has been developed in order to import those XML files into PLCverif translating them into STL code.
- The coverage of Siemens programs was increased both for STL and SCL. More built-in functions from TIA portal were included.
- Support of latest Theta version was included. Theta is being actively developed and new releases have been launched. In order to keep up with the latest improvements, PLCverif has been adapted to correctly parse Theta output. Currently, PLCverif supports Theta v2.21.0.
- The intermediate model Control Flow Instance had the limitation that it could not be used with dynamic indexing arrays. However, Theta supports this feature and PLCverif has been adapted to generate Theta programs with dynamic indexes.
- The grammar implemented in PLCverif to parse Siemens PLC programs has been extended to include partial support of Schneider PLC programs.
- Upgrade to Java 11. PLCverif was originally developed in Java 8. However, in order not to lose support and to be able to run the latest versions of some model checkers (Theta), it was needed to upgrade to Java 11.

ON-GOING CHALLENGES AND DEVELOPMENTS

Simplification of Numeric Variables

As observed in the large code base of CERN industrial PLC systems, one of the main challenges to perform PLC model checking is the state-space explosion originated by the inclusion of numeric variables. PLCverif represents input variables as non-deterministic in the intermediate model. This means that a 16-bit integer is going to have $2^{16} \approxeq 7 \cdot 10^4$ possible values that the model checker needs to explore.

There exist some techniques to handle this type of variables, such as counterexample-guided-abstraction refinement (CEGAR) [7] or Satisfiability Modulo Theories (SMT). However, other approaches to directly simplify the PLCverif intermediate model are under investigation, highly improving the performance of the NuSMV model checker [8].

Iterative Verification

It is common practice to have different modules within PLC projects (see [9] for an example). Some of these projects

[6] https://gitlab.com/plcverif-oss/cern.plcverif
[7] https://gitlab.com/plcverif-oss/cern.plcverif.plc.step7
[8] https://gitlab.com/plcverif-oss/cern.plcverif.gui
[9] https://gitlab.com/plcverif-oss/cern.plcverif.cli

[10] TIA Portal Openness API.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV042

are too large to be verified by PLCverif yet. However, if the program is split into parts, the verification cases are smaller and can be successfully executed. If it would be possible to combine all the results together, it would be feasible to verify these large programs. To this end, different compositional verification approaches have been analysed but no generic method has been found yet to be applied to PLC projects.

Nonetheless, other abstraction techniques are under consideration, such as abstracting away the different modules. With this approach, a verification case is executed with all the functions abstracted away (all their outputs are going to be non-deterministic). If the property is satisfied, the original program satisfies that property too. On the other hand, if it is violated, one needs to check if the abstracted functions can produce the outputs leading to the violation. If it is not possible, a function is concretized (it comes back to its original form) and a new verification case is executed. This process is continued until a feasible counterexample is found or until all functions are concretized (coming back to the original program). Different strategies and improvements can be investigated for this method.

Counterexample Analysis

When a program is verified and the result is a violation of the property, a counterexample is given by PLCverif. If the program is composed of several variables and they interact with each other (see [9] for an example), the counterexample is going to be large. Therefore, it will be difficult to analyse what part of the code made the property fail.

Some investigations have been done in this direction in order to point the user to possible locations in the code that have an impact on the final assertion. This way, the user would not need to go through all the code but just focus on the highlighted parts.

Other

Since the release of PLCverif, there has been some progress in the development of more efficient model checkers. As already explained previously, the latest version of Theta has been integrated into PLCverif. However, although CBMC is efficient, it is a SAT-based model checker that uses few abstraction techniques. Therefore, an SMT-based model checker like ESBMC [10] could improve the performance of CBMC.

CONCLUSION

This paper presented the latest developments of the PLCverif platform to formally verify PLC programs. The developments can be grouped into two main lines of work:

- Promoting and easing the use of PLCverif by making it open source, by supporting more PLC manufacturers (i.e. Schneider Electric), and by guiding the user to the root cause of an issue when a property is violated (counterexample analysis).
- Improving the performance of the verification time by simplifying numerical variables without loosing meaningful information and by implementing an iterative verification process allowing to verify even more complex applications.

All the different developments presented in this paper are in different stages of maturity and are in the pipeline to be included into PLCverif. In addition some new developments will be carried on to support Schneider safety programs and to integrate new model checkers such as ESBMC.

REFERENCES

[1] E. B. Viñuela, D. Darvas, and V. Molnár, "PLCverif Reengineered: An Open Platform for the Formal Analysis of PLC Programs," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 21–27, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-MOBPP01.

[2] D. Darvas, B. Fernández Adiego, A. Vörös, T. Bartha, E. Blanco Viñuela, and V. M. González Suárez, "Formal verification of complex properties on PLC programs," in *Formal Techniques for Distributed Objects, Components, and Systems*, ser. Lecture Notes in Computer Science, E. Ábrahám and C. Palamidessi, Eds., vol. 8461, Springer, 2014, pp. 284–299, ISBN: 978-3-662-43612-7. DOI: 10.1007/978-3-662-43613-4_18.

[3] A. Cimatti *et al.*, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," in *Computer Aided Verification*, E. Brinksma and K. G. Larsen, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 359–364, ISBN: 978-3-540-45657-5. https://nusmv.fbk.eu/

[4] R. Cavada *et al.*, "The nuXmv Symbolic Model Checker," in *Computer Aided Verification - 26th International Conference*, A. Biere and R. Bloem, Eds., ser. Lecture Notes in Computer Science, vol. 8559, Vienna, Austria: Springer, Jul. 2014, pp. 334–342, ISBN: 978-3-319-08866-2. DOI: 10.1007/978-3-319-08867-9.

[5] T. Tóth, Á. Hajdu, A. Vörös, Z. Micskei, and I. Majzik, "Theta: a Framework for Abstraction Refinement-Based Model Checking," in *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*, D. Stewart and G. Weissenbacher, Eds., 2017, pp. 176–179, ISBN: 978-0-9835678-7-5. DOI: 10.23919/FMCAD.2017.8102257.

[6] E. Clarke, D. Kroening, and F. Lerda, "A tool for checking ANSI-C programs," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, K. Jensen and A. Podelski, Eds., ser. Lecture Notes in Computer Science, vol. 2988, Springer, 2004, pp. 168–176, ISBN: 3-540-21299-X. https://www.cprover.org/cbmc/

[7] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-Guided Abstraction Refinement," in *Computer Aided Verification*, E. A. Emerson and A. P. Sistla, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2000, pp. 154–169.

[8] I. D. Lopez-Miguel, B. Fernandez Adiego, J.-C. Tournier, J. A. Rodriguez-Aguilar, and E. Blanco Viñuela, "Simplification of Numeric Variables for PLC Model Checking," in *19th ACM-IEEE International Conference on Formal*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV042

Methods and Models for System Design (MEMOCODE '21), Nov. 2021, ISBN: 978-1-4503-9127-6/21/11. DOI: 10.1145/3487212.3487334.

[9] B. Fernandez Adiego, I. D. Lopez-Miguel, and J.-C. Tournier, "Applying model checking to highly-configurable safety critical software: the SPS-PPS PLC program," presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper WEPV042, this conference.

[10] M. R. Gadelha, F. R. Monteiro, J. Morse, L. C. Cordeiro, B. Fischer, and D. A. Nicole, "ESBMC 5.0: An industrial-strength C model checker," in 33rd *ACM/IEEE Int. Conf. on Automated Software Engineering (ASE'18)*, New York, NY, USA: ACM, 2018, pp. 888–891. DOI: 10.1145/3238147.3240481.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV043

CERN CONTROLS CONFIGURATION SERVICE – EVENT-BASED PROCESSING OF CONTROLS CHANGES

B. Urbaniec, L.Burdzanowski, CERN, Geneva, Switzerland

Abstract

The Controls Configuration Service (CCS) is a core component of the data-driven Control System at CERN. Built around a central database, the CCS provides a range of client APIs and graphical user interfaces (GUI) to enable efficient and user-friendly configuration of Controls. As the entry point for all the modifications to Controls system configurations, the CCS provides the means to ensure global data coherency and propagation of changes across the distributed Controls sub-systems and services. With the aim of achieving global data coherency in the most efficient manner, the need for an advanced data integrator emerged.

The "Controls Configuration Data Lifecycle manager" (CCDL) is the core integration bridge between the distributed Controls sub-systems. It aims to ensure consistent, reliable, and efficient exchange of information and triggering of workflow actions based on events representing Controls configuration changes. The CCDL implements and incorporates cutting-edge technologies used successfully in the IT industry. This paper describes the CCDL architecture, design and technology choices made, as well as the tangible benefits that have been realised since its introduction.

INTRODUCTION

The Controls Configuration Service (CCS) is a core component of CERN's Control system, serving as a central point for the configuration of all Controls sub-domains. CCS ensures that the data provided to other services is done in a coherent and consistent way. CCS is used by a diversified group of users, including installation teams (configuring Controls hardware), equipment experts (configuring processes and applications), and Accelerator operators. All CCS users interact with the service at various points in time, to verify or define appropriate configurations.

The service is built around a centralised Oracle database server. To minimise downtime of the system and risks of negative impact to the users, the server is deployed in a cluster as 2 redundant nodes, providing 99.9% availability. The data stored in the CCS database (CCDB) may be accessed via a dedicated high level web-based editor - Controls Configuration Data Editor (CCDE) [1]. At the same time the service also provides advanced Java and Python REST APIs which allow users to efficiently configure, modify and maintain configuration data in a programmatic way.

The CCS service has been an integral part of the Controls system for many years. The first version was created in the late 80s, during operations of the Large Hadron Collider's predecessor - The Large Electron-Positron Collider (LEP). Since that time the service has evolved and been consolidated multiple times. The last renovation started 4 years ago [2] to match the CCS technology stack with technologies widely used in the software industry. All CCS components are based on Java (currently version 11) and the Spring framework. From the Spring framework, Spring Boot has been selected as a solution to establish a common architecture among all the applications in a simplified and unified way. As mentioned, CCS also provides a high-level web interface - CCDE. The CCDE is based on the AngularJS framework (provided by Google) and is augmented with a web components framework developed in-house and encapsulating common functionality and integration with CERN services such as SSO. Communication between the Java back-end and the AngularJS front-end is implemented using the REST architectural pattern.

CERN CONTROLS CONFIGURATION SERVICE

As a core Controls service, the CCS must exhibit a high level of availability. Even though CCS downtime does not directly impact beam operation, it severely limits the means to verify or modify core system configurations. To provide the highest possible availability and quality of service, each CCS component is implemented with some degree of redundancy. For example service-side processes are stateless and deployed in a multi-node set-up. In the rare case of a failure for one of the nodes, the system remains operational without impacting users. Advanced monitoring and notification mechanisms continuously check the consistency and status of all service components and send alerts in case of any abnormality. This allows service managers to react before the CCS becomes unavailable.

Since its inception, the CCS has evolved regularly, in-synch with the significant evolution of CERN's accelerator complex and multiple sub-systems. During the last 30 years, new advanced accelerators and major upgrades have triggered a need for a more sophisticated and powerful configuration platform. The global Controls architecture is realised as a layered system, reflected in CCS configuration domains, which can be simplified as follows:

- Low-level aspects covering kernel driver configurations and hardware types.
- Front-End Computers (FEC) with their modules (of hardware types, mentioned above), on which,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV043

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

different acquisition and configuration processes are running.

- Software processes which represent physical Devices deployed in specific Accelerators, together with definitions of their interfaces, known as "Device Classes".
- Role-based authorisation schemes, both for access to the aforementioned Device processes and to high-level applications.
- Configuration of Device data acquisitions and logging, to have data available for further analysis.

CONTROLS CHANGES

Every change to Controls configuration, no matter how small, may impact one or more other Control system components. Due to the overall Controls complexity, manual adaptations in related sub-systems to account for changes, are highly time consuming and error prone, and risk to result in discrepancies between different Controls sub-systems. It is therefore crucial, to capture changes, notify related services, and even trigger specific actions, in an automatic way.

To ensure that changes are accepted and parsed correctly, they are treated as named events which may occur during the configuration lifecycle. Each such event may be acknowledged, analysed, and propagated to appropriate services to trigger necessary actions. Once such operations are complete, the user that triggered the change event should be notified of the outcome such as success or failure (including relevant details). The behaviour described above falls into the software architecture paradigm known as an "Event Driven Architecture" and aims to address all implications provoked by change requests.

The evolution of physical Controls hardware requires corresponding adaptations at various layers of the Control system. One example is the meta-data describing the programmatic interface of the equipment ("Device Classes", mentioned above) which must naturally evolve following the hardware. This process of a Device changing Class is called "migration". It is a critical event in the Controls configuration life-cycle, as it may impact a number of core Controls sub-systems such as LSA[3] (for settings management), NXCALS[4] (for time-series data logging), UCAP (for data processing), and end-user applications. To control this impact and minimise risks of inadvertent changes, the "Controls Configuration Data Lifecycle manager" (CCDL) is used.

A typical example of such a change, would be an evolution of a hardware component, such that the meta-data describing the component's properties needs to be updated with respect to its data type (e.g. from a *scalar integer* to a *double array*). Such a change needs to be properly propagated to other Controls sub-systems to assure operational continuity e.g. consistent application of settings or data acquisition. The window within which such changes are carried out (both in terms of underlying hardware and software) is normally limited to either Accelerator technical stops or shutdown periods.

To ensure the correct propagation of changes within the CCS and across related Controls sub-systems, a sub-component of CCDL, the migration "orchestrator" is employed. The orchestration engine reacts to the named configuration change events and processes them by orchestrating the corresponding changes in all related Controls sub-systems. This processing must be done in a logical order, reflecting how the sub-systems depend on one another. End-users are provided with live insights into the event processing and are also notified about any problems. The outcome of each distinct step in the migration processing chain is recorded as a status event. Status events are also injected back into the processing system for use by the orchestrator for optional conditional processing. This solution is deployed in production and Table 1 shows the number of processed events in 2020.

Table 1: 2020 Events Generated Based on User Actions

TabEvent	Occurrences
Device migration (change of Class)	21k
Device attribute changed	280k
Device added	100k
Device deleted	90k
NXCALS subscription changed	250k
NXCALS subscription added	360k
NXCALS subscription deleted	130k

Considering the long history of the CCS and the need to cope with the complexity of the Accelerator Controls domain, the CCS architecture and design are tailored to solve a wide variety of possible problems in a configurable and data-driven manner. Below, the core CCS components are described, together with specific examples of an event-based definition of business use-cases as seen from the perspective of Controls users.

ARCHITECTURE

The main component of the event-based processing of changes in the system is the CCDL - Controls Configuration Data Lifecycle manager. The manager acts as an orchestration and integration point amongst core CCS components (CCDA – Controls Configuration Data API, CCDE – Controls Configuration Data Editor) and a limited number of clients using direct database access with dedicated schemas or PL/SQL APIs. The CCDL is composed of two main components: CCLC (the Lifecycle

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV043

manager) and the so-called "integrated Device migrator". Supporting components of the architecture include:

- Apache Kafka – a distributed data store optimised for ingesting and processing streaming data in real-time.
- Oracle AQ (Advanced Queuing) – a proprietary feature of Oracle databases, which delivers database-integrated message queuing functionality.
- System events – representing configuration changes, are modelled as simple tuple objects, generated at the persistency (database) layer.

System events, modelled as data tuples, consist of meta-data such as operation types, timestamps, aggregation topics (i.e. database transaction ID), as well as the event data (i.e. configuration value before and after the change). The events are designed to be sufficiently rich for the processing system to minimise the need for querying additional data. At the database level the events are stored in log tables, structured with attention to data partitioning and indexing. In normal operations of the system, events are processed continuously which minimises the number of physical reads at the level of the database, thus contributing to overall performance operation of the system. In case when historical events need to be processed, indexing and table partitioning at the level of event type and creation timestamp lowers latency when accessing the data.

The decision to use Apache Kafka was driven by a frequent need for a loosely coupled integration of 3rd party clients with CCS (e.g. other Controls sub-systems). System events are fed to Kafka and made readily available to all subscribed clients, including internal clients like CCLC. This solution enabled a plug-and-play model (see Fig. 1), where external clients may integrate only at the level of Kafka data-objects (key-value pair tuples or JSON objects), or if necessary use a complete CCLC client API, tailored to a specific client domain (e.g. NXCALS or UCAP).

CONCLUSIONS

Before CCDL was put in place, many of the configuration updates described above needed to be performed manually by a limited number of system experts. The manual nature meant that users had to request changes well in advance, such that experts of each of the possibly impacted Controls sub-systems had sufficient time to internally verify what would be the impact, and if necessary, prepare appropriate steps to mitigate risks and ensure a global consistency of the distributed configuration elements. This situation required careful planning and coordination to minimise errors and discrepancies between Controls sub-systems.

The introduction of CCDL helped to considerably reduce support and manual interventions. All changes can now be executed directly by the users, at any moment, without a need for specific actions from the various Controls sub-system experts. Feedback gathered during the last 3 years since the automatic Device migration process has been in place, shows that the work of the users has been simplified and became more efficient. At the same time, thanks to the fully automated process, the time when the global Control system configuration is in an inconsistent state, has been reduced to a minimum.

Reduction of the support and automatization of the configuration process is an example of a continuous evolution of the system. The increasingly dynamic nature of the CERN Accelerator Controls manifested by a growing number of data-driven components, indicates a need for software solutions which in place of customised use-case specific code, rely on more abstract and agile solutions. One possible solution is to deliver a platform which allows users to define complex high-level events based on atomic events generated by the system. While the atomic events represent fundamental changes occurring in the system (e.g. a new Device created), the complex event may be used to indicate patterns of behaviour. For

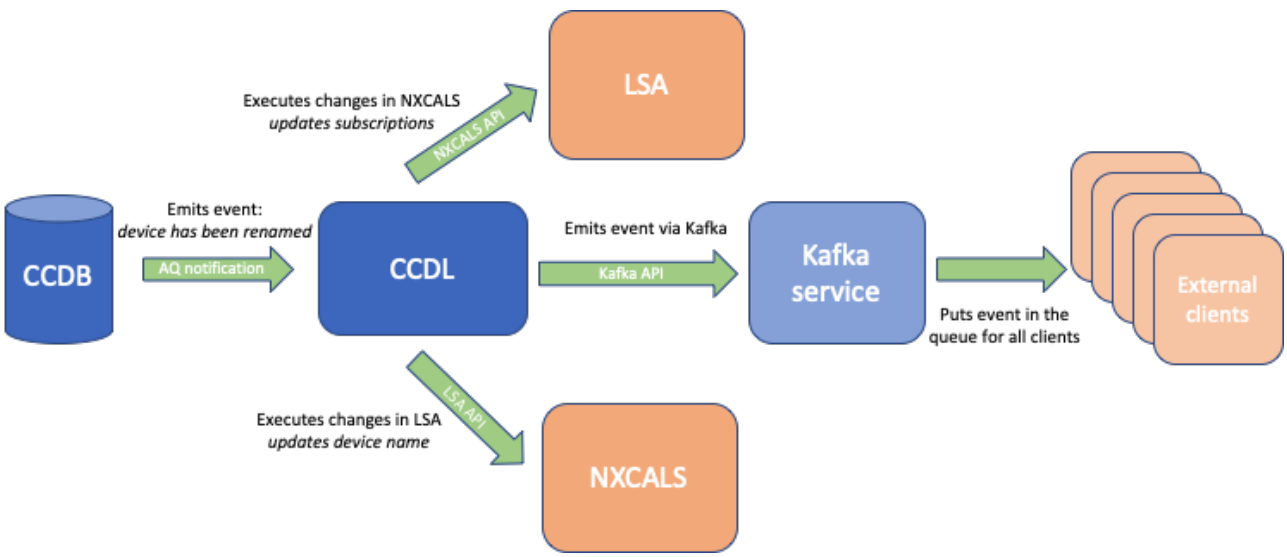


Figure 1: Diagram of event processing with CCDL.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV043

example, a consecutive set of new Device created and Device renamed events may represent a complex event such as swapping of Devices by name while retaining their associated configuration intact. Discovery of such patterns may be achieved with Complex Event Processing engines, for example using Esper CEP framework [5]. The CEP approach provides users with a high degree of freedom in covering different use cases, including dynamic injection of new types of events, without noticeably increasing the amount of support needed from various related Controls sub-system experts.

In the mid-term perspective we plan to integrate ESPER CEP engine with CCDL and consequently expose means to define complex events and EPL (Event Processing Language) queries through a high-level graphical user interface as a part of CCDE. This will provide all the users of the Controls system configuration the means to define new events and give a possibility to look for patterns of events occurring in the system. With such extensions as well as described earlier stream processing engine like Apache Kafka and high-level API like CCDA, we aim to establish practical means to realise a business intelligence solution tailored to the needs of the CERN Accelerator controls and its users, while remaining agnostic to the CERN specific concepts.

REFERENCES

[1] L. Burdzanowski *et al.*, "CERN Controls Configuration Service - a Challenge in Usability", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 159-165. doi:10.18429/JACoW-ICALEPCS2017-TUBPL01

[2] L. Burdzanowski and C. Roderick, "The Renovation of the CERN Controls Configuration Service", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 103-106. doi:10.18429/JACoW-ICALEPCS2015-MOPGF006

[3] D. Jacquet, R. Gorbonosov, and G. Kruk, "LSA - the High Level Application Software of the LHC - and Its Performance During the First Three Years of Operation", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper THPPC058, pp. 1201-1204.

[4] J. P. Wozniak and C. Roderick, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", presented at *the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19),* New York, NY, USA, Oct. 2019, pp. 1465-1469. doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[5] Esper FAQ – EsperTech, https://www.espertech.com/esper/esper-faq

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV044

LESSONS LEARNED MOVING FROM PHARLAP TO LINUX RT

Cédric Charrondiere, Odd Oyvind Andreassen, Diego Sierra Maillo Martinez, Joseph Tagg, Thomas Zilliox, CERN, Geneva, Switzerland

Abstract

The start of the Advanced Proton Driven Plasma Wakefield Acceleration Experiment (AWAKE) [1] facility at CERN in 2016 came with the need for a continuous image acquisition system. The international scientific collaboration responsible for this project requested low and high resolution acquisition at a capture rate of 10Hz and 1 Hz respectively. To match these requirements, GigE digital cameras were connected to a PXI system running PharLap, a real-time operating system, using dual port 1Gbps network cards. With new requirements for a faster acquisition with higher resolution, it was decided to add 10Gbps network cards and a Network Attached Storage (NAS) directly connected to the PXI system to avoid saturating the network. There was also a request to acquire high-resolution images on several cameras during a limited duration, typically 30 seconds, in a burst acquisition mode. To comply with these new requirements PharLap had to be abandoned and replaced with NI Linux RT.

This paper describes the limitation of the PharLap system, and the lessons learned during the transition to NI Linux RT. We will show the improvement of CPU stability and data throughput reached.

INTRODUCTION

A plasma wakefield is a type of wave generated by particles travelling through a plasma. By harnessing these wakefields, accelerating gradients hundreds of times higher than those produced in current radiofrequency cavities can be achieved [2], allowing for more compact accelerators. AWAKE is a proof of principle experiment that aims to demonstrate this in a scalable way, sending proton beams through plasma cells to generate these fields, which subsequently accelerate electrons to high energy over a short distance.

One important observable is the shape and position of the proton beam halo along the beamline, that must be acquired in real-time. To handle the 10Hz image acquisition on 10 cameras simultaneously it was decided to use a PXI running PharLap.

Due to the lack of supported drivers, issues with timing and performance, the system was later upgraded to NI Linux RT to benefit from its flexibility.

Motivation

The cameras are positioned along the beam line for several purposes (Fig. 1). The cameras that are on the virtual laser diagnostic line are used to measure the characteristics of the laser used to initiate the plasma. Other cameras are used to image the path of the laser and proton beams to align them in the plasma cell. Finally, some cameras are positioned to visualize the low energy electron beam at different points of its path before it is accelerated by the proton generated wakefield in the plasma cell.

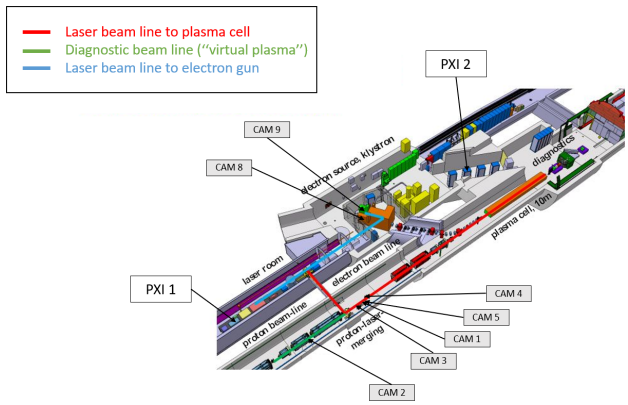


Figure 1: Camera locations in the Awake Experimental Area.

HARDWARE TOPOLOGY

The AWAKE camera acquisition system acquires images from ethernet based, digital GigE cameras. The system publishes resampled images (resampling factor 5x5) at 10 Hz and publishes the full-size images on an SPS extraction event (once per AWAKE cycle). The basic system topology is shown below (Fig. 2)

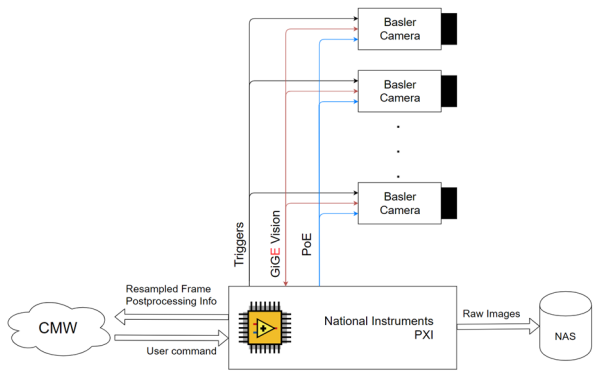


Figure 2: System topology.

Each camera is connected to the PXI system and powered by a PoE (Power over Ethernet) module. The triggering is handled through the FPGA as shown in the hardware architecture (Fig. 3)

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV044

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

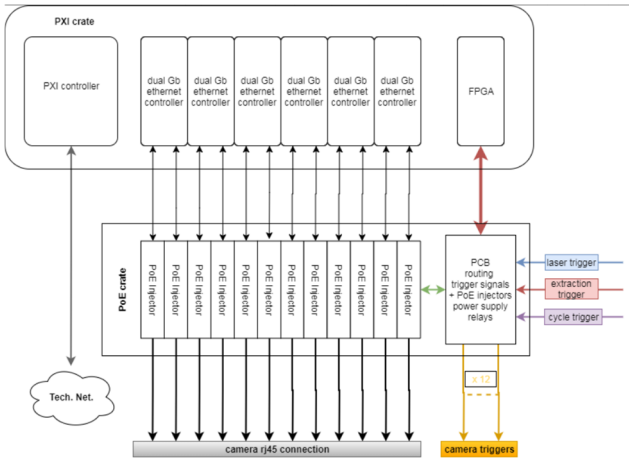


Figure 3: Hardware architecture.

Performance

Two OSes support the new 10Gbps card: NI Linux RT and Windows. We compared the CPU usage on both, using CPUs with relatively close performances, a NI PXIe-8840 for Windows and a NI PXIe-8861 for NI Linux RT. The CPU usage in Windows was around 50% at idle due to th Windows and CERN services running in the background. In production, with cameras acquiring, the PXI was oscillating between 97-99%, resulting in multiple frame losses. With NI Linux RT, the usage of the CPU is around 0% at idle and between 50 to 70%, depending on the number of cameras plugged into the PXI. This increase in available CPU power drastically reduced the frame losses and increased the stability of the system.

Network Attached Storage (NAS)

Publishing the raw images on the CERN network is not possible as its bandwidth is 1 Gbps and the current setup would require 4 Gbps. To allow the AWAKE team to acquire raw images at that rate in the specified amount of time (referred to as burst acquisition) a NAS is directly linked to the PXI's 10 Gbps card. The NAS used is a HP PROLIANT DL380 GEN9 running Linux.

OPERATING SYSTEM

PharLap

The main advantage of PharLap is that it is a lightweight RTOS (Real Time Operating System) with few software layers, which makes it robust and stable while maintaining determinism. This advantage is also a weak point in terms of debugging because when a RT application crashes, the whole OS stops responding, and a reboot of the device is required. In addition, because it does not implement memory virtualization, very little can be fixed at runtime without rebooting, requiring careful memory management. National Instruments will stop its support of PharLap in 2025 [3] with some new modules already not supported on this platform, forcing the community to stop using it for new projects or major upgrades.

In the case of AWAKE, a PXIe-8238 10Gbps ethernet card was required to log data to a NAS at high speed. This card is not supported by PharLap, which motivated us to change the OS.

Another issue is the timing synchronization daemon, SNTP 1.2, based on SNTP (Simple Network Time Protocol), that does not work properly on PharLap. When SNTP is synched, we measure a continuous oscillation of about +/- 0.02 seconds around the correct time. After a few hours (2 to 20+), we measure an abrupt offset of about 4 minutes, followed by oscillations of +/− 0.2 seconds.

Linux RT

As the CERN accelerator complex infrastructure is UNIX based, having an acquisition system based on a NI Linux RT distribution makes its integration into existing services and databases easier. Contrary to PharLap, NI Linux RT (based on the OpenEmbedded platform) ships with built in tools for compiling, debugging, administrating and monitoring our device by default. The behaviour of the built-in NTP timing service works, and debugging via ssh (secure shell) is more convenient since this can be done while the system is running.

Remote access and troubleshooting are relevant to most of our projects as they often are installed in inaccessible areas. As an added bonus, other languages such as Python are also available, opening new possibilities.

In the AWAKE application, we were wary of potential performance issues due to a saturation of the network bandwidth but were not able to measure it. Thanks to the default monitoring tools available in the NI Linux RT distribution, we were finally able to quantify it and fix one of the major issues of the system, which was caused by too many clients connected to the system simultaneously.

In addition to the different monitoring tools, we also benefit from all the different native logging systems. The ethernet logger allowed us to quantify the behaviour of each camera's communication, especially when they were exposed to too much radiation.

The main drawback of NI Linux RT is its backward compatibility; some drivers are not, and may never be, supported.

INTEGRATION

Timing

The different timestamps from the camera's driver, such as the frame arrival timestamp, are based directly on the PXI clock. Because PharLap does not have a trustworthy built-in NTP daemon, there is a permanent clock drift, which requires every timestamp to be corrected. NI Linux RT's NTP client on the other hand works well, which helps with camera timestamping.

With regards to the PXI time, Linux RT is more user-friendly than PharLap. While it is possible to correct in PharLap, Linux RT freed us from this constraint.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV044

Triggering

To sync the image acquisition with the AWAKE laser and other SPS events, a trigger is generated by the AWAKE timing system. It is wired to a CERN central timing receiver card (CTRp) [4] [5], an FPGA-based custom timing card manufactured at CERN. This card allows us to subscribe to every timing event of a specific accelerator, in our case the SPS, and to forward a specific event trigger to all the cameras.

On the early version of the NI Linux RT (up to version 20.6) some VISA functions were not working properly with the CTRp card, such as the subscribe to PXI event (Enable_PXI_VISA_event.vi). We should therefore not take for granted that custom PXI cards that work on PharLap will work out of the box on NI Linux RT.

Vendor Libraries

In general, when using hardware, software and firmware from NI, compatibility and maintenance is not an issue. However, as the years have passed, and PharLap has gone from being a mature to an obsolete platform, even vendor specific compatible drivers have become difficult to find. For the NI Linux RT based systems, the opposite has been the case. Here many of the libraries and drivers have either been ported to the platform in the last few years, or are in the process of being implemented. This poses a challenge for the end user when maintaining the system, leaving you in a situation where some hardware is supported in one environment and not the other. Many of the major hardware families have equivalent replacements available, but they are often not "plug-in" replacements, and might require different signal converters and cables, running at different voltages. This has led to some caution when upgrading systems procured in the 2015 to 2021 timeframe, carefully having to anticipate whether the hardware would have compatible libraries both on PharLap and Linux RT when designing new control system.

In the case of the AWAKE camera acquisition system, the choice was driven by the necessity of higher throughput to cover additional cameras, and a higher level of stability in terms of time drift. The "Lessons learned" moving to NI Linux RT, has been that not everything is a drop-in replacement from the vendor side, and sometimes small hardware changes or features introduced in newer versions, can impact the whole system and should be carefully studied.

3rd Party Libraries

Most libraries on NI based systems are installed using a specialized tool called MAX (Measurement and Automation Explorer). This is the case both for PharLap and NI Linux RT, however the underlying technology for the two platforms is significantly different.

For PharLap, the installation is done simply by formatting the target, priming it with a base operating system and installing drivers by copying them to the right location on disk. There are no convenient ways to keep track of what is installed on which target. To upgrade a driver you need to install the appropriate driver bundle on your development system, which is often versioned differently from what you see on the target. This often leads to confusion and maintenance overhead. To keep track of the libraries installed on the target, it is necessary to document every action taken during the course of development.

For NI Linux RT, the environment is based on UNIX, and in addition to the standard toolchain distributed via a dedicated package manager (OPKG for NI Linux RT), NI have added a management tool called SaltStack, that takes care of monitoring what is installed and if there are upgrades available. In addition, multi-target support has been added, so one can now distribute upgrades to multiple targets in parallel.

This means that system replication is easier for smaller systems on PharLap, but large-scale system administration is by far simpler on NI Linux RT. For third party libraries, this is a huge advantage, allowing custom developed code to be managed side by side with the vendor's own drivers.

Network Booting

One of the main requirements for systems installed in and around the accelerators at CERN, is the ability to remotely recover and re-install the system in case of failure. This was typically done via PXE (Pre-eXecution Environment), however in the later years UEFI (Unified Extensible Firmware Interface) has become more common.

For PharLap, NI uses traditional PXE boot while NI Linux RT NI uses UEFI [6]. Both technologies can be deployed at CERN, but UEFI has more modern capabilities and integrates better into the current infrastructure.

Compilation Process, Debugging and Error Management

When developing 3rd party drivers and libraries such as the CERN middleware and timing libraries, the ability to test and validate the implementation is imperative.

In PharLap, because it is based on an older Windows NT kernel, all custom development requires the Visual Studio development environment, and a PharLap-compatible runtime.

To debug libraries, you first have to test the library under Windows and then rigorously test all functionality on PharLap. The problem is that PharLap does not have any proper debugging tools, and the error messages it produces are often obscure or too generic, forcing you to add extensive debug information as part of the implementation (such as printing information along the execution of the code).

On NI Linux RT, you have all the standard GCC and UNIX tool chains available, and debugging can easily be done in a comfortable way, saving both development and maintenance time.

Compilation Tools

As mentioned, PharLap relies on the Visual Studio toolchain to compile 3rd party software. Unfortunately, the development of a dedicated Visual Studio runtime for PharLap was stopped at version 2010, and Microsoft, the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV044

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

owner of Visual Studio stopped its support for the runtime in its development environment in 2020 (official EOL was 2015 but extended EOL was pushed back top 2020). Furthermore, Visual Studio 2010 is not compatible with Windows 10 and beyond, making compilation of source code for PharLap based systems even more complicated [7].

For NI Linux RT, the standard gcc or g++ toolchain can be used, even on the target itself, and both development and validation is straight forward. If in addition you leverage automated builds through continuous integration and modern DevOps solutions (such as those provided by GitLab), any upstream change to a dependent source can be caught, compiled, and tested immediately as it is released. This has proven to be a huge time saver in terms of consolidation and maintenance.

CONCLUSION

Switching from PharLap to NI Linux RT brought performance improvements, reducing both the general resource and CPU usage. The UNIX environment simplified library and system administration. It also allowed features that were missing from PharLap such as the 10 Gbps network drivers and the NAS.

The next challenges will be to increase the image post-processing speed in order to increase the resolution of the cameras. As a result of the operating system change, we can now benefit from newer FPGA cards dedicated to vision analysis, allowing us to offload some calculations from the CPU.

REFERENCES

[1] https://home.cern/science/accelerators/awake

[2] https://home.cern/science/engineering/accelerating-radiofrequency-cavities

[3] https://www.ni.com/content/dam/web/pdfs/phar-lap-rt-eol-roadmap.pdf

[4] J. Serrano, P. Álvarez, D. Domínguez, and J. Lewis, "Nanosecond Level UTC Timing Generation and Stamping in CERN's LHC", in Proc. 9th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'03), Gyeongju, Korea, Oct. 2003, paper MP533, pp. 119-121.

[5] O.O.Andreassen *et al.*, "Evaluation of timing synchronization techniques on NI CompactRIO platforms", ICALEPCS 2019, New York, Oct. 2019, paper WEPHA27.

[6] O.O.Andreassen *et al.*, "Integrating COTS Equipment in the CERN Accelerator Domain", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA026.

[7] https://docs.microsoft.com/en-us/lifecycle/products/visual-studio-2010

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

DATA-CENTRIC WEB INFRASTRUCTURE FOR CERN RADIATION AND ENVIRONMENTAL PROTECTION MONITORING

Adrien Ledeul*, Catalina Cristina Chiriac, Gonzalo de la Cruz, Gustavo Segura, Jan Sznajd
CERN, Geneva, Switzerland

Abstract

Supervision, Control and Data Acquisition (SCADA) systems generate large amounts of data over time. Analyzing collected data is essential to discover useful information, prevent failures, and generate reports. Facilitating access to data is of utmost importance to exploit the information generated by SCADA systems.

CERN's occupational Health & Safety and Environmental protection (HSE) Unit operates a web infrastructure allowing users of the Radiation and Environment Monitoring Unified Supervision (REMUS) to visualize and extract near-real-time and historical data from desktop and mobile devices. This application, REMUS Web, collects and combines data from multiple sources and presents it to the users in a format suitable for analysis.

The web application and the SCADA system can operate independently thanks to a data-centric, loosely coupled architecture. They are connected through common data sources such as the open-source streaming platform Apache Kafka and Oracle Rdb. This paper describes the benefits of providing a feature-rich web application as a complement to control systems. Moreover, it details the underlying architecture of the solution and its capabilities.

INTRODUCTION

Radiation protection and environmental monitoring are fundamental aspects of the CERN Safety Policy. CERN's occupational Health & Safety and Environmental protection (HSE) Unit conducts a program in charge of monitoring the radiological and environmental impact of the organization. The aim is to ensure workplace safety for CERN employees and visitors, minimize the environmental impact of CERN, and provide regulatory authorities with comprehensive reports.

In order to achieve these objectives, a geographically distributed and heterogeneous set of instruments is continuously measuring the nature and quantity of ionizing radiations produced by the accelerators, possible contamination as well as conventional environmental parameters.

Radiation and Environment Monitoring Unified Supervision (REMUS) [1], based on WinCC Open Architecture (WinCC OA) [2] is the Supervision, Control And Data Acquisition (SCADA) system controlling this infrastructure. It is accessed from various control rooms across the Organization and has more than 200 active users.

At the time of writing, REMUS is interfacing 86 different types of devices. It contains 850 000 tags, manages 120 000 alarms and handles a throughput of 25 000 Input/Output operations per second. REMUS archives roughly 80 billion measurements per year.

One of the main challenges of such a system is to provide comprehensive yet accessible means to extract and exploit the data. REMUS itself allows users to display a large variety of synoptic views and control panels designed for the operation in control rooms. However, such user interfaces are not the most suitable for data extraction at a higher level of abstraction, and typically require physical access to terminals in a protected network.

Two use cases are particularly challenging to handle. The first one is that CERN radiation and environmental protection experts are in charge of transforming the data generated by REMUS system into business-specific reports. Such reports are used for further internal analysis and to consolidate CERN's communication with the host states and with the general public. The second one is to provide the CERN's Fire and Rescue service and other emergency response teams with access to near-real-time data on remote terminals. This is particularly useful at CERN, where installations for environmental monitoring are geographically scattered. The installation of dedicated monitoring screens is not always possible at the location where the access to the data is needed.

This paper describes the approach taken to make radiation and environmental monitoring data accessible to third party applications as well as why web technologies were chosen for the presentation of this data.

A DATA-CENTRIC APPROACH

This section introduces the approach taken for the consolidation and homogenization of the data layer.

The Data-Centric Manifesto

The data-centric mindset, as opposed to the application-centric one, considers the data to be the permanent assets, and the applications the temporary ones. The key principles, as expressed in the Data-Centric Manifesto [3], can be summarized as follows:

- Data is the key asset.
- Data is self-describing.
- Data is stored in non-proprietary formats.
- Data access control and security is the responsibility of the data layer itself.

This approach suits well REMUS use case, as radiation and environmental protection data must be kept for an indefinite amount of time, as requested by regulatory authorities. On the other hand, the applications publishing

* adrien.ledeul@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

and consuming the data change regularly, driven by technological evolution.

The key point is to think and design the data-set before designing the applications fulfilling the requirements. The concepts of a data-centric mindset applied to the CERN Radiological and Environmental Monitoring System are illustrated in Fig. 1.

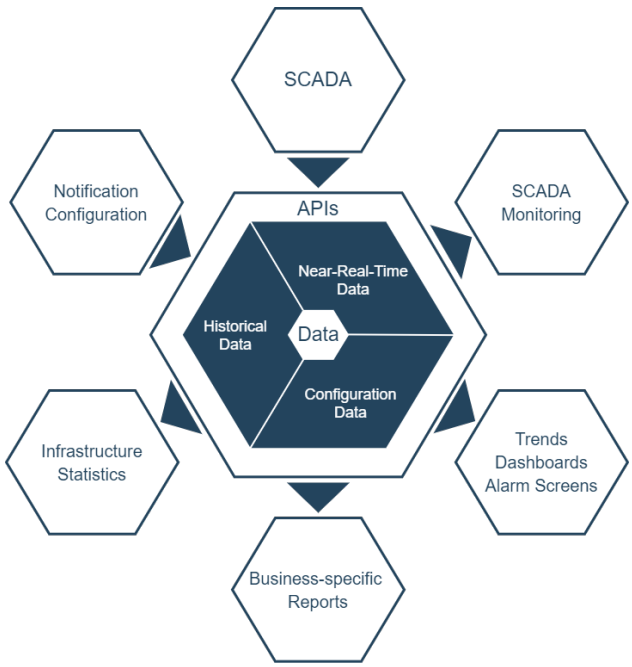


Figure 1: Data-Centric principles applied to REMUS. The center of the diagram represents all the data associated with radiation and environmental protection monitoring at CERN. The external blocks represents applications consuming and producing data through APIs.

The first steps taken in REMUS towards a data-centric approach were to consolidate and complete the data layer, as well as to unify the applications access to the multiple data sources of the control system through a common API.

REMUS DATA PIPELINE

This section describes the REMUS data flow, from the instruments to the data layer used by visualization tools.

Overview

REMUS is in charge of the data acquisition of over 5500 measurement channels, generating about 2600 data points per second in total. The acquired data is processed in parallel in two different ways, following the Lambda architecture [4]. On the one hand, data is processed in batch for long-term archiving. On the other hand, data is processed as a stream to allow immediate and continued exploitation [5]. The aim is to provide a data layer that can be homogeneously accessed through a data access API for historical and near-real-time data queries. The full data pipeline is illustrated in Fig. 2.

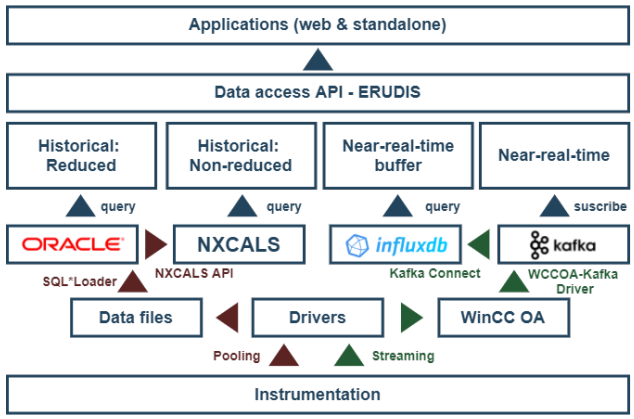


Figure 2: REMUS Data Pipeline. Applications consume data aggregated by the data access API. Data is fetched from various data sources with different latencies and time resolutions. Oracle and NXCALS are used for historical data; Kafka and InfluxDB for near-real-time data.

Batch Processing

REMUS fetches the data buffered in instruments' internal memories at a configurable polling interval. This data is written into files that are then injected in Oracle Rdb's partitioned tables using SQL*Loader [6] processes. A subset of the data is then transferred to the Next CERN Accelerator Logging Service (NXCALS) [7], based on Hadoop [8], where it is kept indefinitely at the highest resolution. The data stored in Oracle Rdb is reduced after two weeks, in order to guarantee low latency access.

Stream Processing

In parallel to the batch processing, REMUS receives continuous data streams from the instrumentation. The streams are transformed by the SCADA and published to a dedicated Kafka [9] topic using an open-source WinCC OA Kafka Driver [10], developed and maintained by CERN's HSE Unit. The topic is then filtered using Kafka Streams [11] to eliminate potential malformed messages. The processed stream is then sent to InfluxDB [12] via Kafka Connect [13] for temporary data retention.

Data Access API

Environment and Radiation Unified Data Integration Service (ERUDIS) is a data access API. ERUDIS provides high-level abstraction of the data sources that can be consumed by third party applications. The architecture of this API is based on Akka, an actor concurrency model framework, and the Alpakka [14] library, designed for handling data streams. These tools simplify the creation of interfaces for unifying heterogeneous data sources, as well as provide means for interactive management of data streams, such as buffering, advanced error handling and automatic redundancy mechanisms.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

REMUS WEB

This section introduces REMUS Web, a web infrastructure that allows users to visualize and extract near-real-time and historical data from desktop and mobile devices. REMUS Web takes advantage of the data pipeline introduced in the previous section to present the data to the user in a format suitable for analysis.

Rationale of the Approach

Web Applications Since its invention at CERN, the World-Wide Web [15] has evolved from a hosting platform for simple and static hypermedia documents to an infrastructure for the execution of complex applications [16]. In the last decades web applications became a popular and widespread solution in the software industry due to the advantages they present over standalone applications.

From the user's point of view, web applications are more accessible. They can be reached through the web browser and do not require complex installation processes or specific hardware configurations. Web applications work on any device capable of running a simple web browser, offering greater portability and cross-platform access. Furthermore, the requirements that end-user workstations must satisfy to run the applications are minimal, since computationally expensive tasks are delegated to the server.

From the developer's point of view, maintenance and update processes of web applications are simpler compared to standalone applications, since the application only needs to be deployed on the servers and no intervention is required on the end-user workstations. Web technologies are the most widely used and the most mature for the development of graphical user interfaces (GUI), which facilitates and speeds up development thanks to the large ecosystem, plugins and community around web development. In addition, web technologies are widely spread in the IT industry and it is easier to recruit professionals with web development skills than with SCADA and control systems skills. According to the 2021 Stack Overflow Developer Survey [17], web developer roles are the most popular on the market.

Progressive Web Applications In recent years the importance and impact of the web has been increasing. Most modern software companies base their business models on offering services through the web. This has meant the consolidation of the web as a service platform and the emergence of new web technologies that have made the line between web applications and native applications increasingly thin. In this model, cross-platform compatibility (desktop, mobile, and tablets) is essential to maximize the reach of the application and to provide the best user experience.

Progressive Web Applications (PWA) [18] provide users an experience on par with native applications. PWAs are a type of web applications that are intended to work on any platform using a standard-compliant browser, including both desktop and mobile devices. PWAs take advantage of recent advancements in web browser technologies such as service workers [19], web app manifests [20], and caching to bring features usually associated with native apps. These features include installability, responsiveness, network independence, re-engageability, and enhanced security. PWAs offer similar features to native and hybrid [21] applications, with a smaller bundle size and fast loading times [22]. Using PWA can significantly reduce the costs and resources required for the development and maintenance of a multi-platform application, since all platforms share the same codebase.

Data Accessibility The aforementioned features solve several of the challenges posed by accessibility of control systems' data. The use of a web application facilitates remote access to the data generated by the SCADA from any desktop or mobile device with a standard-compliant web browser. This is especially useful when operators must carry out interventions in remote facilities where there is no access to the control system terminals. Furthermore, REMUS Web is built as a PWA, allowing operators to access the data generated by every equipment from their mobile device. REMUS Web provides QR codes for every instrument connected to REMUS, which can be printed and attached to the instruments. In this way, when operators need to intervene on an instrument, they can access the instrument data in real time by scanning its QR code. This mechanism also allows operators in the field to access data from devices that do not have any integrated display.

Technological Stack

The technologies and tools used in the development of REMUS Web are listed below. These technologies have facilitated and accelerated the development of the tool, as well as improved its maintainability and scalability.

Front-end HTML, CSS and JavaScript are the standard technologies used for the development and design of web applications and therefore are the base technologies on which REMUS Web front-end is built. Additionally, high-level tools and libraries facilitate application development.

Most modern web applications are built on JavaScript frameworks for GUI development, which simplify the application development and maintenance processes. Currently, there are three major JavaScript frameworks that dominate the market: React [23], Angular [24], and Vue.js [25]. These frameworks have different characteristics and the decision of which one is the best option will depend largely on the circumstances of the project to be developed [26].

In the case of REMUS Web, we considered React to be the preferred option. React is a free and open-source JavaScript library for building GUIs which is developed and maintained by Facebook. React is flexible, efficient and declarative. According to the 2021 Stack Overflow Developer Survey [17], React is the most commonly used web framework and the most wanted by developers. At the performance level, React is fast and light. It offers a similar

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

performance to that of its direct competitors [27]. Since React is a UI library and not a complete framework, it offers a lot of flexibility and freedom to decide the architecture, tools and libraries used to build the application. Unlike other frameworks that use their own directives and templating languages, React's code is essentially pure JavaScript, which makes the learning curve shorter.

React has built-in functionalities for state management that are suitable for simple projects. However, the need for a more sophisticated state management solution arises as the complexity of the application increases and it becomes necessary to share state between different components. Redux [28] is the most widespread solution and the one used in REMUS Web. Redux is a predictable state container for JavaScript applications based on the Flux [29] architectural pattern.

Since one of the main goals of REMUS Web is to facilitate the study and analysis of the measured data, the use of a powerful and flexible charting library is of utmost importance. To this end, REMUS Web uses Highcharts [30]. Highcharts is a consolidated JavaScript SVG-based multi-platform charting library. It is flexible, well adapted to desktop and mobile devices and offers a wide variety of charts.

Real-time access to SCADA data away from the central control room is vital to assess and intervene quickly in critical situations. REMUS Web uses the WebSocket protocol [31] to establish a full-duplex real-time communication channel between the client and the server. In this way, the new data generated by the SCADA system is streamed to the client with a very low latency.

Back-end REMUS Web back-end is based on Java [32] and Spring Boot [33]. Spring Boot is an open source, microservice-based Java web framework. It is an extension of the well known Spring Framework [34], simplifying the set-up and configuration of standalone Spring applications and taking an opinionated view of the Spring platform and third party libraries to facilitate the development process.

Architecture

Figure 3 shows the architecture of REMUS Web. The application is deployed on the CERN Platform-as-a-Service (PaaS) infrastructure, based on RedHat OpenShift [35]. OpenShift is a container orchestration platform optimized for web applications. It allows building, testing and deploying web applications without provisioning and maintaining dedicated servers for each application. OpenShift runs the applications in Docker [36] containers.

The front-end of the application runs on the users' web browser and communicates with the back-end in two different ways depending on the nature of the information to be transmitted. WebSockets are used for the transmission of real-time data such as measurements or alarms. They are also used to send commands from the client to the server in order to start, stop or modify the parameters of the transmitted data flow. The rest of the communication between the client and the server is done through HTTP [37] requests.

The back-end of the application follows a classic three-layer architecture. The presentation layer exposes the REST and WebSocket APIs that the front-end consumes to interact with the back-end. The business layer contains the domain logic that drives the core functionalities of the application. Finally, the data layer is responsible for interacting with different data sources in order to persist and fetch data.

The data layer is based on Spring Data and ERUDIS. Spring Data is used to access and modify the data that is strictly owned by REMUS Web, such as reports, dashboards, or user preferences. In addition, it is also used to access the metadata of the SCADA system. On the other hand, ERUDIS is used to access both the historical and near-real-time data of the instrumentation connected to REMUS.

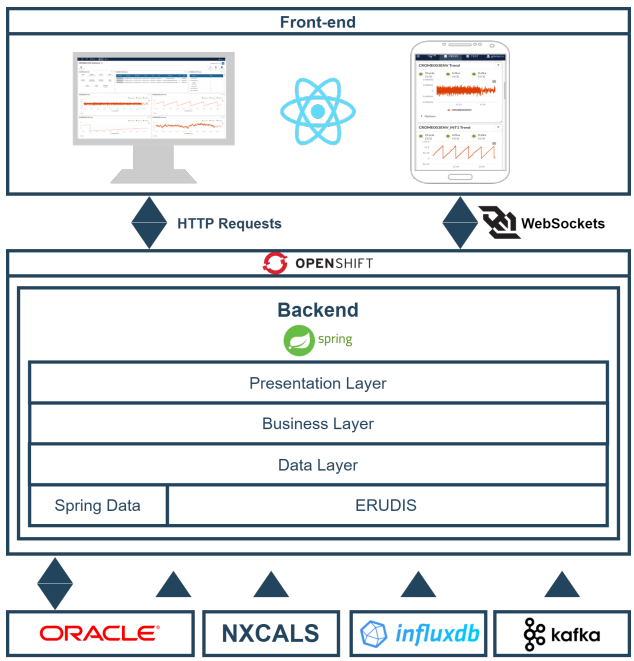


Figure 3: REMUS Web Architecture.

Functionalities

This section introduces some of the most important functionalities that are included in REMUS Web.

Notification Configuration Users can configure personalized e-mail and SMS notifications based on events produced by the SCADA. Users can select the notification sources, the event that triggers the notification (e.g. an alarm goes on or off) and the notification lifetime.

SCADA Monitoring The application allows the generation of custom reports to monitor the current and past status of the SCADA system. It is especially useful for analyzing incidents and to carry out investigations.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

Near-real-time Alarm Screen REMUS alarm screen is accessible directly from the web and displays the generated alarms in near-real-time. This eliminates the need to connect to the SCADA remote terminals in order to access the alarm screen, being accessible from any desktop or mobile device with internet access.

Near-real-time Trends REMUS Web allows to access and plot (see Fig. 4) both historical and near-real-time SCADA data. Users can fetch data from multiple data sources and define various parameters on the data to select. These parameters include time period, data resolution, and filters based on the alarm and fault status or the working mode of the instrument that generated the measurements.

Figure 4: REMUS Web Trends.

Domain-specific reports REMUS users require domain-specific reports for radiation and environmental protection. Such reports are used for internal analysis and to report to CERN's host states authorities. REMUS Web includes tools to generate these reports. Use cases include the extraction of aggregated radiation data and generation of the hyetographs (see Fig. 5).

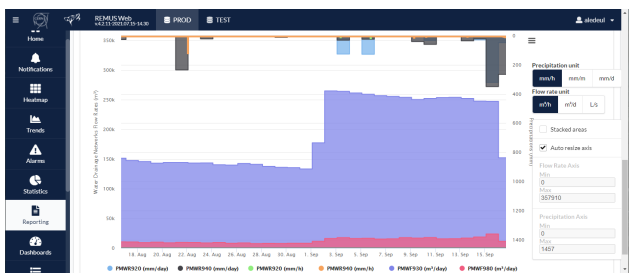


Figure 5: REMUS Web Reports. This screenshot shows a generated hyetograph, allowing experts to analyse the impact of rain on water flow rates.

Dashboards REMUS Web includes a powerful tool for creating and visualizing dashboards. The tool makes a wide variety of widgets available to users. Users can compose their own custom dashboards by combining different types of widgets. The layout and size of the widgets can be modified using a drag-and-drop interface. In addition, REMUS Web automatically generates a dashboard for each of the instruments connected to REMUS. Such dashboards show real-time data from this particular instrument, its alarms, and its parameters, as shown in Fig. 6.

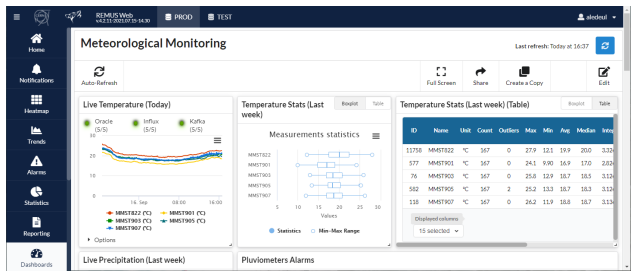


Figure 6: REMUS Web Dashboards.

Metadata Statistics REMUS Web allows access to the complete inventory of REMUS entities such as measurement channels, instruments, and events. Users can filter the data using multiple criteria and perform data aggregations to extract statistics and charts, as shown in Fig. 7.

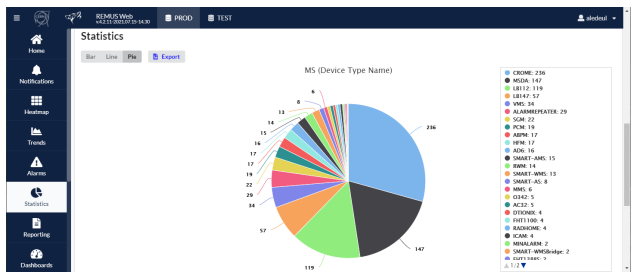


Figure 7: REMUS Web Statistics.

CONCLUSION

Control Systems' Human-Machine-Interface went a long way, from push buttons and wired lamps to modern SCADA graphical user interfaces. The fast evolution of data architectures and web technologies is an opportunity to go one step further in the accessibility of SCADA's presentation layer.

CERN's Radiation and Environmental protection SCADA extended its data extraction and analysis capabilities, exploiting a modern data-centric architecture and web technologies.

ACKNOWLEDGMENT

We would like to thank all the members, past and present, of the REMUS Project team as well as colleagues from Radiation Protection, Environment, Beams and Information Technology groups for their fundamental contribution to the success of the REMUS Web project.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] A. Ledeul, G. S. Millan, A. Savulescu, B. Styczen, and D. V. Ribeira, "CERN supervision, control and data acquisition system for radiation and environmental protection," in *Proceedings of the 12th International Workshop on Emerging Technologies and Scientific Facilities Controls (PCaPAC'18)*, 2019, p. 248.

[2] Wincc oa, https://www.winccoa.com

[3] Data-centric manifesto, http://datacentricmanifesto.org

[4] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 2785–2792.

[5] A. Ledeul, A. Savulescu, G. S. Millan, and B. Styczen, "Data streaming with apache kafka for cern supervision, control and data acquisition system for radiation and environmental protection," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, 2019.

[6] J. Gennick and S. Mishra, *Oracle SQL* Loader: the definitive guide*. " O'Reilly Media, Inc.", 2001.

[7] J. Wozniak, C. Roderick, and S. R. WEPHA163, "Nxcals-architecture and challenges of the next cern accelerator logging service," in *17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA*, 2019.

[8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, Ieee, 2010, pp. 1–10.

[9] J. Kreps, N. Narkhede, J. Rao, *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011, pp. 1–7.

[10] Github repository for winccoakafkadrv, https://github.com/cern-hse-computing/WCCOAkafkaDrv

[11] Kafka streams, https://kafka.apache.org/documentation/streams

[12] Influx db, https://www.influxdata.com

[13] Kafka connect, https://www.confluent.io/product/connectors

[14] Alpakka, https://doc.akka.io/docs/alpakka

[15] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The world-wide web," *Communications of the ACM*, vol. 37, no. 8, pp. 76–82, 1994.

[16] S. Casteleyn, F. Daniel, P. Dolog, and M. Matera, *Engineering web applications*. Springer, 2009, vol. 30.

[17] Stack overflow developer survey 2021, https://insights.stackoverflow.com/survey/2021

[18] Progressive web apps, https://web.dev/progressive-web-apps

[19] M. Kruisselbrink, A. Russell, J. Song, and J. Archibald, "Service workers 1," W3C, Candidate Recommendation, Nov. 2019. https://www.w3.org/TR/2019/CR-service-workers-1-20191119/

[20] A. Gustafson, "Web app manifest - application information," W3C, W3C Note, Mar. 2021. https://www.w3.org/TR/2021/NOTE-manifest-app-info-20210324/

[21] A. Khandeparkar, R. Gupta, and B. Sindhya, "An introduction to hybrid platform mobile application development," *International Journal of Computer Applications*, vol. 118, no. 15, 2015.

[22] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive web apps: The possible web-native unifier for mobile development," in *International Conference on Web Information Systems and Technologies*, SciTePress, vol. 2, 2017, pp. 344–351.

[23] React, https://reactjs.org

[24] Angular, https://angular.io

[25] Vue.js, https://vuejs.org

[26] E. Wohlgethan, "Supportingweb development decisions by comparing three major javascript frameworks: Angular, react and vue. js," Ph.D. dissertation, Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[27] S. Krause, Js framework benchmark - round 8, https://stefankrause.net/js-frameworks-benchmark8/table.html

[28] Redux, https://redux.js.org

[29] Flux application architecture, https://facebook.github.io/flux

[30] Highcharts, https://www.highcharts.com

[31] I. Fette and A. Melnikov, The websocket protocol, 2011.

[32] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[33] Spring boot, https://spring.io/projects/spring-boot

[34] Spring framework, https://spring.io/projects/spring-framework

[35] A. Lossent, A. R. Peon, and A. Wagner, "Paas for web applications with openshift origin," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 898, 2017, p. 082 037.

[36] D. Merkel *et al.*, "Docker: Lightweight linux containers for consistent development and deployment,"

[37] R. Fielding *et al.*, Hypertext transfer protocol–http/1.1, 1999.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV047

UPGRADING ORACLE APEX APPLICATIONS AT THE NATIONAL IGNITION FACILITY

A. Bhasker, R. D. Clark, R.N. Fallejo
Lawrence Livermore National Lab, CA, USA

Abstract

As with all experimental physics facilities, NIF has software applications that must persist on a multi-decade timescale. They must be kept up to date for viability and sustainability. We present the steps and challenges involved in a major Oracle APEX application upgrade project from Oracle APEX version 5 to Oracle APEX version 19.2. This upgrade involved jumping over 2 major versions and a total of 5 releases of Oracle APEX. Some applications - that depended on now legacy Oracle APEX constructs required rearchitecting, while others that broke due to custom JavaScript needed to be updated with compatible code. This upgrade project, undertaken by the NIF Shot Data Systems (SDS) team at Lawrence Livermore National Laboratory (LLNL), involved reverse-engineering functional requirements for applications that were then redesigned using the latest APEX out-of-the-box functionality, as well as identifying changes made in the newer Oracle APEX built-in "plumbing" to update custom-built functionality for compatibility with the newer Oracle APEX version. As NIF enters into its second decade of operations, this upgrade allows for these aging Oracle APEX applications to function in a more sustainable way, while enhancing user experience with a more modernized GUI for existing and future Oracle APEX webpages.

INTRODUCTION

The National Ignition Facility (NIF) made history when it first fired all 192 of its laser beams at a single point inside the vast, spherical target chamber in 2009. Now, over 10 years and more than 2,700 shots later, NIF still dominates the world of high-energy lasers by a factor of 10 and continues to help the Laboratory achieve its mission objectives, offering unparalleled laser performance and precision [1]. As NIF enters into its second decade of operations, it must continue to maintain its various software applications in a sustainable and viable way. Two such applications that the Software Data Systems (SDS) team at NIF recently upgraded are the *Production Optics Reporting and Tracking* and the *Shot Planner* applications. At the time of this undertaking, these applications were running on Oracle APEX version 5.0. Oracle APEX is a web-application development tool for the Oracle databases [2], that supports utilization of JavaScript and JQuery components as well. In this paper, we discuss the steps involved in upgrading these NIF applications from Oracle APEX version 5.0 to Oracle APEX version 19.2 – a jump spanning 2 major versions and a total of 5 releases of Oracle APEX.

The Oracle APEX Applications

The original *Production Optics Reporting and Tracking (PORT)* and the *Shot Planner* applications were developed in very early versions of Oracle APEX when NIF was in the process of being built, over a decade ago. The PORT application covers a wide domain of capabilities – ranging from reporting and analysis to task-scheduling and optics installation. The Shot Planner application functions as a scheduling app, providing reports and forms to plan the sequencing of shots on NIF. At the time of the Oracle APEX upgrade, both these applications were running on legacy Oracle APEX themes and utilizing certain Oracle APEX constructs (such as AnyChart/Flash Charts [3]) that are no longer supported by the latest Oracle APEX versions.

METHODOLOGY

The PORT application comprised of ~500 application pages, and the Shot Planner application comprised of ~200 application pages at the time of this upgrade. The following are the steps that the team undertook as a part of the Oracle APEX upgrade for these applications (Figure 1).

1. Identify all Obsolete Application Pages
2. Set up an APEX 19.2 environment for testing
3. Fix Forward Incompatibility
4. Deploy Pre-Upgrade Fixes
5. Line Up Post-Upgrade Fixes
6. Oracle APEX Upgrade Utility Recommendations
7. Upgrade and Deploy Post-Upgrade Fixes

Figure 1: Methodology followed.

Identify Obsolete Webpages

The SDS team worked with the various users to identify any application pages that were no longer used. Given the age of these applications, several application functionalities were no longer required as the program evolved with time. Identifying these application pages gave the team an opportunity to prune out the deprecated functionalities. To be sure that a functionality could be deprecated, all visits of the various application pages on these applications were recorded for a period of one year before beginning the actual upgrade. To achieve this, a new Oracle database table was created to record this information from the APEX_WORKSPACE_ACTIVITY_ LOG view [4].

A Sandbox Environment with Oracle APEX 19.2

The next step was setting up a test environment with the target Oracle APEX version 19.2 installed. Copies of the application were imported into this *sandbox* environment. The team meticulously tested all the functionalities on each Oracle APEX application page and identified features that were no longer working in the new upgraded environment. This was done by comparing each application page in the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

sandbox against the actual *production* version of the application page and documenting the results.

Categorize/Fix Forward Incompatibility

The comparison test between the application pages on the sandbox v/s production environments helped identify whether a page could be upgraded to Oracle APEX version 19.2 as-is, or if some additional work needed to be done to bring it to a functioning state. Broadly speaking, the comparison test resulted in pages falling in one of the following three categories:

- The application page could be upgraded as is, i.e., no additional work required.
- The application page rendered initially but some smaller functionalities no longer worked. Example: List of Values (LoV) now require both an explicit return and display value while they did offer backward compatibility for only defining one value to be used for both display and return.
- The application page did not render at all due to some browser incompatibility. This typically meant a JavaScript or JQuery error.

Fortunately, the majority of the application pages belonged to the first category, where the pages could be upgraded as is. However, there were still many high impact application pages on both the applications that needed to be fixed before the production environment could be upgraded to Oracle APEX 19.2. At this point, *two* important determinations need to be made. First, determining the cause behind the application pages not behaving exactly the same as they did in the old environment. We found that the cause was one of the two (Figure 2):

- The application page had an Oracle APEX feature implemented that was deprecated and no longer compatible with version 19.2. These were typically minor fixes that mostly required small changes in code.
- Or, there was custom JavaScript/JQuery either embedded in the application page itself or in a custom plug-in used by that application page. The fixes for these ran the full gamut, from being small code fixes like updating the *html/css* class names being referred to in JavaScript code, to reverse engineering and redesigning an entire application functionality using the latest Oracle APEX 19.2 constructs like Interactive Grids.

Second, determining whether the fix can be applied in the current production environment version *before* upgrading to the new version (Figure 3). For instance, fixes involving certain unsupported APEX features could be made in the current version. An example would be rewriting the List of Values (LoV) definitions to include both return and display values. An example where the fix could not be applied before upgrading would be an application page using complex custom Javascript that is not compatible with the Javascript versions native to the latest Oracle APEX version and was redesigned using the latest Oracle APEX 19.2 out-of-the-box constructs.

By the end of this step, the sandbox environment had been used to develop mitigations for all types of broken application pages, and the sandbox environment was a functional equivalent of the production environment.

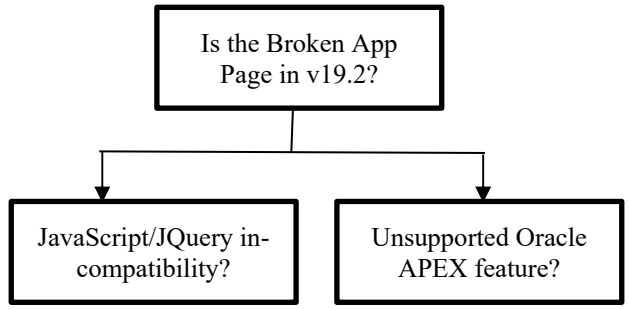


Figure 2: Page forward incompatibility causes.

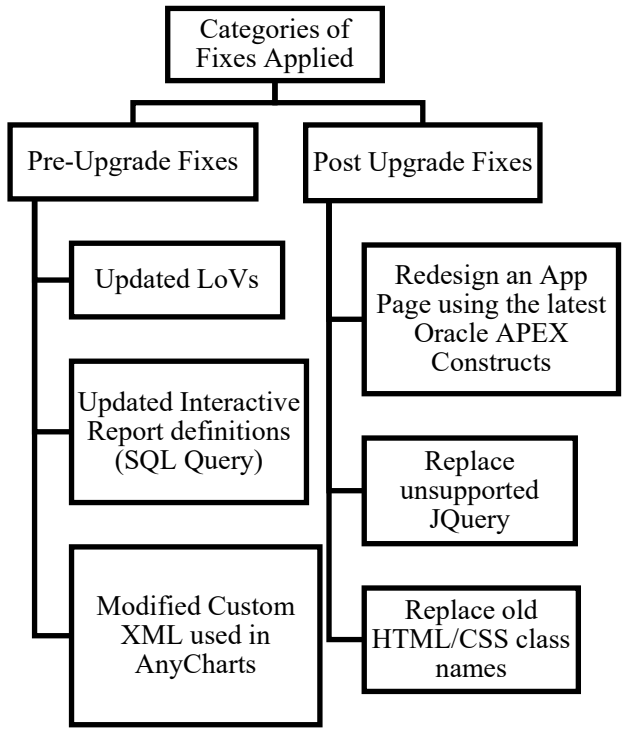


Figure 3: Examples of fixes for a smooth upgrade.

Apply Pre-Upgrade Fixes

Any identified fix that could be applied to the APEX 5.0 version environment was deployed, and the applications were once again imported to the sandbox environment. A second round of testing confirmed that the specific pages with the pre-applied fixes now worked correctly in the Oracle APEX 19.2 environment.

Export the Post-Upgrade Fixes

Any re-architected pages using new Oracle APEX constructs like Interactive Grids or Oracle JET charts were exported from the *sandbox* environment and applied as a software release immediately after upgrading the production environment to Oracle APEX 19.2. Another example of a post-upgrade fix includes several pieces of custom JavaScript code and plugins that use the *html/css* class names to add custom features to regular Oracle APEX constructs like Interactive Reports and Classic Reports. For instance,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV047

the APEX Interactive Report class is now called 'a-IRR-table' in APEX 19.2; previously this was called 'apexir_WORKSHEET_DATA'.

Upgrade Utility

Oracle APEX has an in-built 'Upgrade Application' Utility which makes recommendations for individual features in pages that can be upgraded. Figure 4 below shows an example of some upgrade categories it auto-suggests. The team also tested the specific application pages listed under these upgrade categories after accepting these upgrades in the sandbox. Some of these upgrades do not result in functional equivalents of the original page. For example, accepting the upgrades to upgrade Tabular Forms (which are still compatible with Oracle APEX 19.2) to Interactive Grids did not result in functioning pages, and required further re-design of the page. Such upgrades were selectively applied during this phase of the upgrade. Other upgrades, such as 'Enable Group By for Interactive Reports' or 'Enable Save Public Reports' were accepted using this utility, as they only enhanced the features on the page.

Upgrade and Deployment

With all the fixes identified, applied and user-tested over the course of several weeks in the upgraded sandbox environment, the upgrade was successfully performed in the production environment, followed by the deployment of the post-release fixes, and then accepting the pre-tested page-wise upgrades from the APEX Upgrade Utility. The entire effort took around 4 months, which including meticulous testing, identifying the causes for various incompatibilities and any redesign efforts.

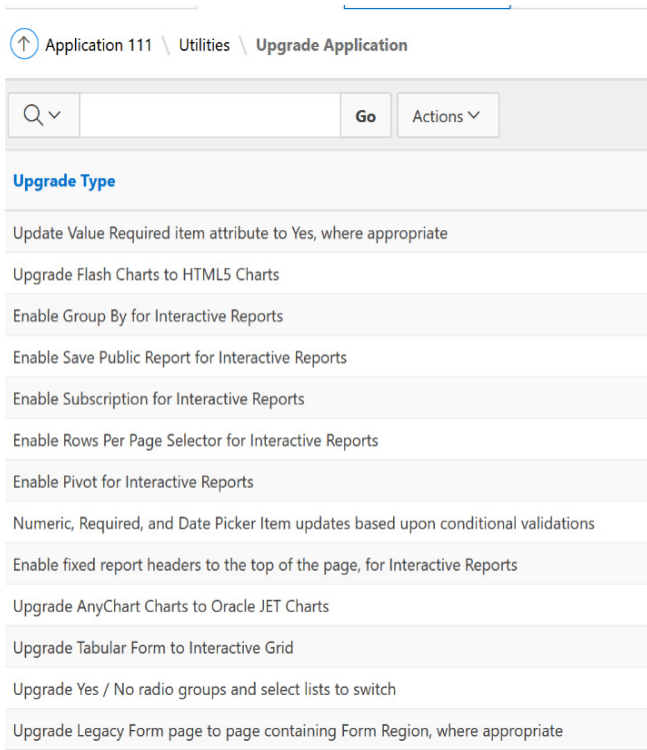


Figure 4: APEX upgrade utility.

NOTABLE LESSONS LEARNED

Some issues encountered in the upgrade that are worth mentioning:

- There were several cases of JQuery incompatibility when running the same application page on the same browser, but on the two different versions of APEX. Example: replacing "$.browser.msie" with "navigator.appname" got several pages working again.
- Oracle APEX has changed the CSS class names of some of its more popular constructs like Interactive Reports. Legacy custom code as well as plug-ins often utilize these.
- Some functionalities were redesigned using Interactive Grid instead of applying complex custom JavaScript to Interactive Reports to implement some *Form* functionalities in older versions of APEX. It was discovered that while the Interactive Grids in Oracle APEX allowed users to save Private and Public reports (same as Interactive Reports do), these saved reports are not exported as a part of an application page export. This is significant because this may impact the Oracle APEX administrator's ability to maintain individual application backups containing all saved reports. Fortunately, the full application export does in fact contain the saved reports.
- The APEX_PLUGIN.GET_AJAX_IDENTIFIER() function can sometimes include hyphens in the unique AJAX-ID generated in the upgraded Oracle APEX version. Some browsers may have compatibility issues with this and throw console errors.
- Some LoVs had invalid values that could only be viewed in the *Page Component* View in Oracle.
- APEX 5.0. The LoV's values could not be viewed in Oracle APEX 19.2's *Page Designer* view at all. These LoV definitions were updated as part of the pre-upgrade fixes as the *Page Component* view is not available in Oracle APEX 19.2.
- Interactive Reports with SQL query definitions of the format *select * from <object>* needed redefinition with explicit column names in the select clause. This was for niche cases where the backend definitions had been updated since the application page was created, and the page accommodated for this in APEX 5.0. However, when ported to the APEX 19.2 environment, it generated errors because it was looking for fields that did not exist in the source object anymore.

CURRENT STATUS AND FUTURE WORK

The upgrade project of the Production Optics Reporting and Tracking and the Shot Planner applications from Oracle APEX version 5.0 to Oracle APEX 19.2 was the first phase of an overarching project to keep it in an easily maintainable state, following recommendations from Oracle on best practices with Oracle APEX. The next phase of the project includes upgrading the Oracle APEX Theme for the two applications to the Oracle recommended Universal Theme. This upgrade to Oracle APEX 19.2 allows for these aging Oracle APEX applications to function in a more

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV047

sustainable and secure way, while enhancing user experience with a more modernized GUI for existing and future Oracle APEX webpages.

ACKNOWLEDGMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-827563.

REFERENCES

[1] National Ignition Facility & Photon Science, https://lasers.llnl.gov

[2] APEX Application Development, www.oracle.com/database/technologies/appdev/apex-what-is.html

[3] Switching from AnyChart to JET Chart, https://docs.oracle.com/en/database/oracle/application-express/19.2/htmdb/switching-AnyChart-to-JET-Chart.html

[4] Creating Custom Activity Reports Using APEX_ACTIVITY_LOG, https://docs.oracle.com/cd/E59726_01/doc.50/e39147/advnc_act_log.htm#HTMDB130

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

FAST MULTIPOLE METHOD (FMM)-BASED PARTICLE ACCELERATOR SIMULATIONS IN THE CONTEXT OF TUNE DEPRESSION STUDIES *

M. Harper Langston†[1], Richard Lethin[1], Pierre-David Letourneau[1], Julia Wei[1]
[1]Reservoir Labs Inc., New York, NY 10012, USA

Abstract

As part of the MACH-B (Multipole Accelerator Codes for Hadron Beams) project, Reservoir Labs has developed a Fast Multipole Method (FMM [1–7])-based tool for higher fidelity modeling of particle accelerators for high-energy physics within Fermilab's Synergia [8, 9] simulation package. We present results from our implementations with a focus on studying the difference between tune depression estimates obtained using PIC codes for computing the particle interactions versus those obtained using FMM-based algorithms integrated within Synergia. In simulating the self-interactions and macroparticle actions necessary for accurate simulations, we present a newly-developed kernel inside of a kernel-independent FMM, where near-field kernels are modified to incorporate smoothing while still maintaining consistency at the boundary of the far-field regime. Each simulation relies on Synergia with one major difference: the way in which particles interactions are computed. Specifically, following our integration of the FMM into Synergia, changes between PIC-based computations and FMM-based computations are made by simply selecting the desired method for near-field (and self) particle interactions.

INTRODUCTION

The majority of numerical approaches for accelerator multiparticle-tracking solve the macroscale problem by employing Particle-In-Cell (PIC) methods [8–14]. These methods incorporate an Eulerian method for solving the necessary equations and Lagrangian techniques to advect particles through the domain (e.g., see Fig. 1).

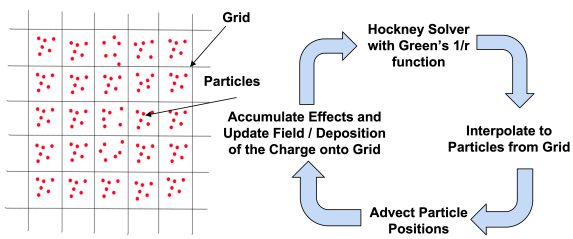


Figure 1: PIC-based Hockney solver. Given a cloud of charged particles, iterate (1) Grid charge deposition; (2) Compute potential; (3) Compute forces at grid points; (4) Compute forces at particle locations.

Since space-charge modeling in high-intensity hadron beams for the accelerator physics community requires scalable and high-fidelity algorithmic approaches, all new computational approaches must **(1)** be inherently multiscale, **(2)** exploit locality, **(3)** reduce expense of non-locality while handling accuracy, **(4)** guarantee high accuracy when needed, and **(5)** handle a variety of complex geometries.

Reservoir Labs' MACH-B (Multipole Accelerator Codes for Hadron Beams) project addresses the above five key elements, maintaining the strengths of PIC codes and approaches while further improving upon some of their weaknesses, allowing domain experts to evaluate and optimize various scenarios for complex high-energy physics experiments. The MACH-B technology is based on both existing and novel mathematical frameworks, providing scalable, high-performance algorithms that will assist in accurately and rapidly computing a variety of complex particle accelerator simulations; specifically, (1) **Fast Multipole Methods (FMM)** and (2) **Boundary Integral Solvers (BIS)**.

Introduction to Fast Multipole Methods

FMM approaches achieve linear scaling by separating near- and far-field interactions (e.g., see Fig. 2) on a spatial hierarchy using tree data structures. As they achieve arbitrary precision at modest cost with straightforward error estimates [1,2,4–6,15–20], FMM techniques are well-suited for problems requiring high accuracy at large scales, such as in particle accelerator simulations.

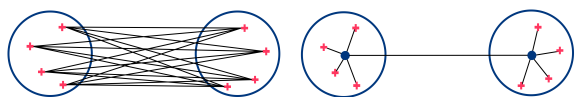


Figure 2: (*Left*): A naive $O(N^2)$ approach for computing the interactions between well-separated sources and targets. (*Right*): Using multipole and local expansions to reduce far-field costs, based on refinement.

FMMs are **inherently multiscale**, separating a regular domain into disjoint sets, using a tree structure to **exploit locality** as well as **reduce the expense of non-locality** through low-rank approximation multipole expansions [1, 4]. FMMs compute the total field at a domain B as the sum of (a) the field due to the sources contained in its near field $\mathcal{N}^B$ and (b) its far field $\mathcal{F}^B$. Contributions from $\mathcal{N}^B$ are computed using direct, dense summations, while contributions from $\mathcal{F}^B$ are obtained by evaluating approximating expansion coefficients, constructed to achieve far-field low-rank approximations at computationally-efficient and provably-accurate levels of accuracy. Through two parameters ((1) for points per smallest grid in the hierarchy and (2) for number of coefficients in the expansions), **high accuracy is guaranteed** [2, 3, 21].

* This work was supported by the U.S. Department of Energy as part of the SBIR Phase I Project DE-SC0020934.
† langston@reservoir.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The FMM uses *upward* and *downward passes* on a hierarchical tree structure, employing multiple operators for converting expansion (multipole and local) coefficients to **achieve optimal $O(n)$ complexity** (See Fig. 3 from [4]).

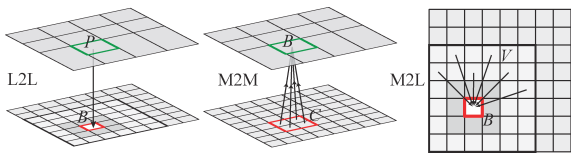


Figure 3: Local to Local (L2L), Multipole to Multipole (M2M) and Multipole to Local (M2L) operators translate coefficients efficiently throughout the FMM algorithm.

INTEGRATING FMM INTO SYNERGIA

In this work, we focus on MACH-B's effort in integrating the FMM algorithm [1, 20] into Synergia [8, 9] for computing particle interactions as compared to existing PIC-based methods. We rely on the STKFMM from [15], itself a generalization of the highly-efficient PVFMM [6] implementation of the kernel-independent FMM [3–5, 19]. We chose this particular implementation of the FMM for two main reasons:

1. *Kernel-Independence*: rather than applying only to the Laplace kernel $(1/r)$, this version is kernel independent (in particular, it can handle any elliptic kernel).
2. *Computational Performance*: both the theory and implementation of this version are among the most performant currently existing.

Prior to the introduction of the FMM, Synergia offered two ways of computing interactions, both based on the Particle-In-Cell (PIC) method: `space_charge_2d_open_hockney()` and `space_charge_3d_open_hockney()`. All rely on Hockney's method (as seen in Fig. 1) and can be summarized as follows: given a cloud of charged particles,

1. *Grid Charge Deposition*: for each cloud particle, "deposit"/interpolate its charge on a regular uniform grid.
2. *Compute Potential*: compute the potential generated by charges from a grid point at all other grid points using the FFT (convolution with electrostatic kernel $1/r$).
3. *Compute Forces at Grid Points*: compute 3D forces at each grid point using finite differences.
4. *Compute Forces at Particles*: interpolate force values from the uniform grid to the particle locations.

The FMM was integrated into Synergia such that changes from native PIC-based routines to FMM-routines are performed with minimal effort; currently, switching between methods requires merely changing one word in the name of the aforementioned functions to: `space_charge_2d_open_fmm()` and `space_charge_3d_open_fmm()`, respectively.

Verifying Correctness of FMM in Synergia

To verify correctness and performance of each implementation, we perform to three different sets of experiments:

- *Test 1*: Compare Synergia's 3D Hockney uniform grid routine potential computations with a naive approach.
- *Test 2*: Compare 3D FMM routine potential computations with a naive approach (same grid as Test 1).
- *Test 3*: Compare 3D FMM, 3D Hockney (Synergia) and naive force computations on arbitrary point clouds.

The first two tests verify the correctness of each implementation, positioning random charges at uniform grid points. The potential at each grid point is computed using a naive $(O(N^2))$ method and constitutes the ground truth against which we compare solutions. We limit ourselves to 32 points per dimension since computations through the naive method become to onerous beyond this point. Because particles are located at grid points, no interpolation is necessary for the PIC codes; hence, expected numerical accuracy for PIC-based methods matches that of the FFT itself(i.e., $\epsilon \approx 10^{-15}$ in double precision) as seen in Table 1.

Table 1: Test 1: Compare Synergia's 3D Hockney routine potential computations with naive approach on uniform grid. The PIC-based routine behaves as expected.

G_{pts} per dim	Total G_{pts}	Rel. Err.
8	512	$4.56 \cdot 10^{-16}$
16	4096	$1.36 \cdot 10^{-15}$
32	32768	$3.78 \cdot 10^{-15}$

Test 2 is analogous to Test 1, except we employ the FMM for computing particle interactions rather than Synergia's PIC-based methods. The FMM is oblivious to charges abeing located at gridpoints. The FMM parameter, p, dictates its accuracy and computational costs; single precision can be expected with $p \approx 8$ and double precision with $p \approx 12$ while the latter takes about twice as long as the former. Table 2, shows results for different values of p, where it is observed that the FMM behaves as expected.

Table 2: Test 2: Compare 3D STKFMM routine potential computations with naive approach on uniform grid (same grid as Test 1). The FMM-based routine behaves as expected with higher accuracy as p is increased.

G_{pts} per dim	Total G_{pts}	Rel. Err. (p=8)	Rel. Err. (p=12)
8	512	$4.95 \cdot 10^{-9}$	$1.91 \cdot 10^{-11}$
16	4096	$2.95 \cdot 10^{-9}$	$9.72 \cdot 10^{-12}$
32	32768	$2.97 \cdot 10^{-9}$	$5.16 \cdot 10^{-12}$

Test 3 compares the accuracy of Synergia's PIC-based methods and the FMM in a general case: a cloud of charged particles. For this purpose, we generate a 3D ensemble of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

charges (positive/protons) containing up to 32768 particles and then compare to the naive solutions in Table 3. The FMM ($p = 8$) achieves full (single) precision while the PIC-based method suffers from significantly lower accuracy due to interpolations and finite difference steps (Step 1, Step 3 and Step 4 from Fig. 1) associated with PIC codes. When particles, are not at grid points (as in Test 1) and forces must be computed, the error resulting from these steps becomes dominant.

Table 3: Test 3: Compare 3D STKFMM, 3D Hockney (Synergia) and Naive force computations on arbitrary point clouds. The FMM preserves its accuracy regardless of the particle distribution, whereas the PIC-based methods suffers from significant numerical errors associated with interpolation and finite differences.

$N_{particles}$	Param. (M/p)	Rel. Err. (PIC/FMM)
4096	32/8	$0.228 / 1.28 \cdot 10^{-7}$
16384	32/8	$0.166 / 2.07 \cdot 10^{-7}$
32768	32/8	$0.141 / 2.66 \cdot 10^{-7}$
4096	64/8	$0.180 / 1.28 \cdot 10^{-7}$
16384	64/8	$0.141 / 2.07 \cdot 10^{-7}$
32768	64/8	$0.123 / 2.66 \cdot 10^{-7}$
4096	128/8	$0.116 / 1.28 \cdot 10^{-7}$
16384	128/8	$0.106 / 2.07 \cdot 10^{-7}$
32768	128/8	$0.0992 / 2.66 \cdot 10^{-7}$
4096	256/8	$0.0587 / 1.28 \cdot 10^{-7}$
16384	256/8	$0.0640 / 2.07 \cdot 10^{-7}$
32768	256/8	$0.0656 / 2.66 \cdot 10^{-7}$

To these results, we note the following observations:

1. In the most general case (e.g., arbitrary particle clouds), the FMM offers greater accuracy and speed compared with PIC-based methods.

2. In special cases, including the case where the underlying solution is smooth (e.g., as the number of particles goes to infinity., i.e., Vlasov), PIC methods may exhibit higher accuracy (see tune depression case below), although such conditions may be difficult to verify.

TUNE DEPRESSION SIMULATIONS

The (x/y) *tune* of a particle is defined as the number of (x/y) transverse oscillations a particle experiences per revolution around a closed-loop accelerator. Tune is heavily-dependent upon the nature of the accelerator lattice, but also depends strongly on the interactions between the particles (self-energy). For instance, researchers at Fermi Lab have shown an interesting relationship between the transverse (x/y) offset of a particle's original position within a Gaussian bunch and its tune. Figure 4 shows the computation of the tune for particles with initial offset between $0\,\sigma$ and $5\,\sigma$ (measured in number of bunch transverse standard deviations: σ) In particular, this figure shows the importance of self-interactions. Indeed, no changes in the tune are observed

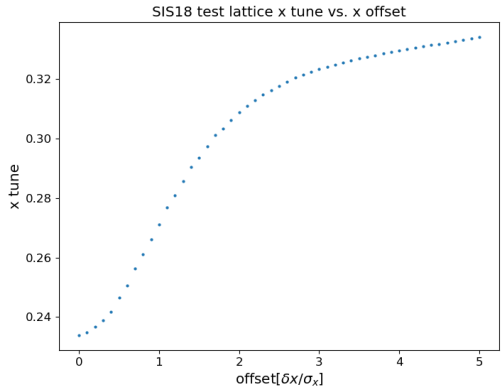


Figure 4: Tune depression (Hz; x-transverse oscillations per turn) vs initial x offset. Results obtained using Synergia PIC routine for computing particle interactions.

when the interactions are neglected, while an interesting relationship is present when they are. This demonstrates the importance of particle-particle interactions in this context and, more generally, for accelerator simulations.

In the context of MACH-B, we study the difference between tune depression estimates obtained in Synergia from PIC versus FMM, adopting the following methodology: first, create a Gaussian bunch of size approximately $10^{-5} \times 10^{-5} \times 10^{4}$ *mm* centered at the origin and containing approximately 10^{10} physical particles (protons) and between 10^{4} and 10^{7} macroparticles depending on the situation (see below for more information about macroparticles). Each macroparticle is randomly assigned a position and momentum following a Gaussian distribution with mean 0 and covariance matrix specific to the lattice, chosen so that the overall shape of the bunch remains constant at each iteration. We introduce "spectator particles" with specific initial offsets to monitor the tune as a function of offset.

After initializing the particle bunch, we set the particles in motion and record the transverse position of each spectator particles after each turn (50 to 500 with 10^{1} to 10^{2} time steps per simulation). We extract the tune as a function of offset by taking the Fourier transform of the x/y transverse position as function of turn. The estimate for the tune is taken to be the dominant frequency identified using the resulting spectrum. Finally, we plot the relationship between the estimated tune and the original offset.

We reproduce the results obtained using PIC codes by using an FMM in its stead. Important differences are observed, however, and this leads to multiple key observations:

1. FMMs can reproduce observed results from PIC codes in the context of tune depression computations.

   (a) FMMs faithfully capture the physical interactions between particles up to numerical accuracy, while PIC methods introduce a large degree of regularization(often desirable).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

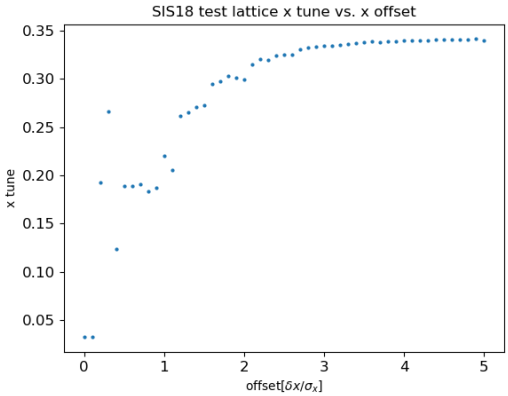


Figure 5: Tune depression (Hz; x-transverse oscillations per turn) vs initial x offset. Results obtained using FMM integrated within Synergia for computing particle interactions. PIC routines introduce more regularization than the FMM resulting in smoother (less noisy) estimates.

2. It is computationally necessary to use macroparticles to represent a particle bunch as the number of particles needed to capture the full dynamics ($> 10^{12}$).

   (a) The use of macroparticles requires the user to use a modified Laplace kernel for representing interactions with the FMM.

   (b) The number of macroparticles and type of modified kernel affects results; i.e., regularization of the singularity at the origin has the largest effects.

These observations are important for understanding the advantages and disadvantages of using FMMs for particle accelerator simulations as detailed below.

FMM vs. Tune Depression Simulations

As previously mentioned and shown in Fig. 5, we successfully reproduced the relationship between tune depression and initial particle offset within a Gaussian bunch using Synergia with an FMMs replacing PIC. However, results obtained using the FMM tend to be more noisy than those obtained using Synergia's native PIC methods.

This phenomenon is mostly due to the inherent regularization associated with PIC methods; the key for computational efficiency here is the use of the FFT, which leads to an $O(N\log(N))$ scaling (N being the number of grid points) rather than a prohibitive $O(N^2)$ scaling in the case of a naive computation. The use of the FFT, although efficient, comes at a cost: the use of a uniform (DFT) grid. The need to go between charges in an arbitrary configuration to (proxy) charges on a regular grid entails the use of interpolation (deposition) further introducing artificial regularization. Indeed, after deposition, two particles that were originally close to each other, and that would thus have exerted large forces upon each other, now find themselves much farther apart and thus exerting much weaker forces.

By contrast, the FMM can compute all interactions between particles with high numerical accuracy without the need for deposition, interpolation, or a regular grid or finite differences, thus avoiding regularization. In addition to preserving high accuracy, FMM costs remains low (i.e., $O(M)$ for M particles). It turns out, however, that this absence of regularization may not always be beneficial.

Indeed, in this specific case (Gaussian bunch), regularization proved beneficial as seen from the qualitative differences between the smoother curve in Fig. 5 and noisier one in Fig. 5. The main reason behind this is that the underlying true (asymptotic as the number of particles goes to infinity) solution for this particular problem is itself smooth. In this context, regularization can capture the underlying physics (one does not lose information by smoothing/regularizing since the solution is already smooth). By contrast, the FMM suffers from strong fluctuations due to small local charge density inhomogeneities that are negligibly small in the asymptotic solution (e.g., Debye screening).

While PIC-based regularization may be advantageous in certain contexts, it is, however, not always appropriate since its success ultimately relies on strong un-verifiable hypotheses regarding the underlying physic. Instead, the FMM offers a fast alternate approach to computing particle interactions without the need for such strong assumptions. While a potential introduction of noise in the estimated solution is possible, this noise can be controlled through the use of macroparticles and modified interaction kernels.

When the charge density is not homogeneous, or when the interactions between multiple bunches are considered, PIC methods are further disadvantageous: the rectangular domain covered by the regular grid needed must contain all particles present in a PIC simulation. However, for an inhomogeneous density, this results in potentially more grid points than particles (many having zero charge), and therefore an FFT cost (proportional to number of gridpoints) much higher than that of the FMM (proportional to the number of particles).

Macroparticles and Incorporation into FMMs

Despite significant advantages of FMMs over PIC methods, it necessary to use macroparticles for large simulations purposes. *Macroparticle* is an all-encompassing term for a family of heuristics, by which the number of particles in a simulation is significantly reduced compared with the number of physical particles.[1] In Synergia, macroparticles are created through a smaller number of particles with larger charges. For instance, to perform a PIC-based simulations in Synergia using 10^{12} physical particles of charge e, but only 10^7 macroparticles, one would simply create a simulation with 10^7 particles having charge $\frac{10^{12}}{10^7}\, e = 10^5\, e$.

Although often appropriate, such simple macroparticles may not always capture physical phenomena appropriately. In this sense, a more suitable way of creating macroparticles

[1] Given their fewer numbers, the interactions between marcoparticles *must* be modified if the solution is to represent a larger number of particles.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

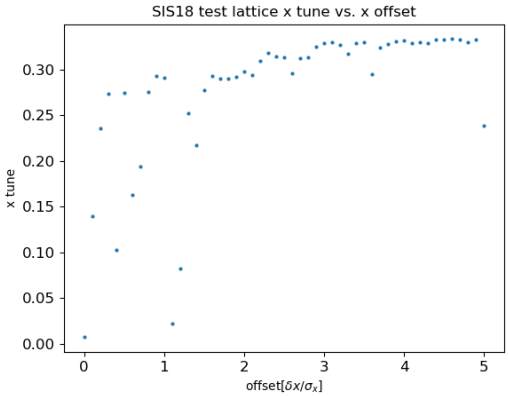


Figure 6: Tune depression (x) vs original x offset for modified interaction kernel ($1/(r+\delta)$) using simulations with $6.4 \cdot 10^6$ physical particles and $6.4 \cdot 10^4$ macroparticles. In Fig. 5, $\delta = 10^{-2}$ was set while here, we set $\delta = 10^{-5}$. Note how less regularization leads to a much noisier outcome.

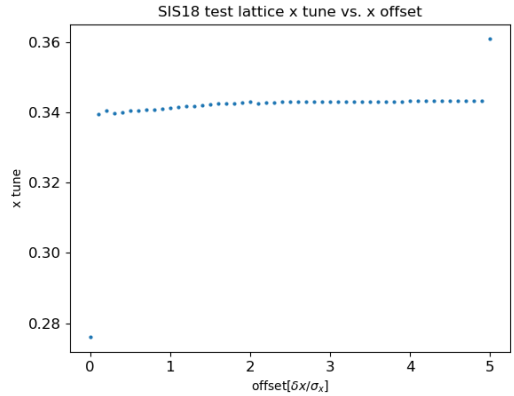


Figure 7: Tune depression (x) vs original x offset for a varying number of physical and macroparticles. In Fig. 6, $N_{physical} = 6.4 \cdot 10^6$, $N_{macro} = 6.4 \cdot 10^6$, whereas here $N_{physical} = N_{macro} = 6.4 \cdot 10^4$. Note how using too few physical particles (or inappropriate macroparticles) fails to capture the appropriate behavior.

is through the use of a *modified interaction kernel*. Modifying the interaction kernel (electrostatic potential $1/r$) by reducing the strength of the singularity at the origin, can be used to introduce regularization as well as to emulate the behavior of the particles in the asymptotic limit (e.g., local averaging, Debye screening, etc.). In MACH-B, we modify this kernel by adding a small positive quantity to the radius, so that the interaction kernel now takes the form: $1/(r+\delta)$, for some $\delta > 0$, mollifying the strong forces at the origin (responsible for noise) while behaving like a regular electrostatic potential far from the origin.

The parameter δ can have a significant impact on the qualitative behavior of the solution for computing the tune depression in the previous section as shown in Fig. 6 as compared to Fig. 5. These figures show computations of the tune depression using the same bunch but with two different interaction kernel δ values. Figure 6 effectively approaches the unregularized case, whereas Fig. 5 represents macroparticles that interact like point particles when far away, but produce much smaller forces when close by. In particular, note the how much smoother the results are when using even a minimal ($\delta = 10^{-2}$) amount of regularization.

Macroparticles are not merely for computational convenience; their appropriate modeling is *necessary* for capturing the physics. For instance, for the tune depression simulations described in the previous section, one could simply try to reduce the number of physical particles rather introducing macroparticles. This, however, fails to generate the desired outcome as shown in Fig. 7 because a bunch with significantly fewer particles will behave significantly differently than a dense bunch. A dense bunch, however, cannot be simulated directly because too many physical particles are required.

CONCLUSIONS AND FUTURE WORK

In MACH-B, we have sucesfully integrated FMMs into the Synergia particle accelerator simulation software suite and have shown the correctness of our implementation and have provided information favoring the use of FMM in many cases, especially when higher accuracy is needed or when the underlying solution may exhibit strong variations.

In our tune depression studies, the advantages of the FMM in this case are salient. Indeed, our FMM algorithm is constructed in such a way that it can handle a much larger family of macro-particle interaction kernels than can PIC codes, allowing the user to introduce various kinds of regularization when appropriate, or to simply do away with any sort of regularization without loss of accuracy. By constrast, PIC codes *always* introduce a significant amount of regularization which nature is quite rigid and difficult to modify. The FMM also offers better scalings than PIC codes in many cases (e.g., multiple bunches).

As we continue to modify our kernels and incorporate them into additional PIC-related codes, we anticipate the FMM libraries will become powerful tools for domain scientists and researchers for study and cross-verification, largely due to their ease of use and modification.

ACKNOWLEDGMENT

Thank you to Drs. Eric Stern and Qiming Lu at Fermilab for helpful conversations and guiding us through Synergia as well as Drs. Robert Ryne and Jean-Luc Vay at LBNL for many very fruitful discussions.

REFERENCES

[1] L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations", *Journal of Computational Physics*, vol. 73,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

no. 2, pp. 325–348, 1987. doi:10.1016/0021-9991(87)90140-9

[2] L. Greengard, "Fast Algorithms for Classical Physics", *Science*, vol. 265, no. 5174, pp. 909–914, 1994. doi:10.1126/science.265.5174.909

[3] L. Ying, G. Biros and D. Zorin, "A Kernel-Independent Adaptive Fast Multipole Algorithm in Two and Three Dimensions", *Journal of Computational Physics*, vol. 196, no. 2, pp. 591–626, 2004. doi:10.1016/j.jcp.2003.11.021

[4] M.H. Langston, L. Greengard, D. Zorin, "A Free-Space Adaptive FMM-Based PDE Solver in Three Dimensions", *Communications in Applied Mathematics and Computational Science*, vol. 6, no. 1, pp. 79–122, 2011. doi:10.2140/camcos.2011.6.79

[5] I. Lashuk *et al.*, "A Massively Parallel Adaptive Fast-Multipole Method on Heterogeneous Architectures", in *Proceedings of ACM/IEEE SC'09*, 2009. doi:10.1145/1654059.1654118

[6] D. Malhotra and G. Biros, "PVFMM: A Parallel Kernel Independent FMM for Particle and Volume Potentials", *Communications in Computational Physics*, vol. 18, no. 3, pp. 808–830, 2015. doi:10.4208/cicp.020215.150515sw

[7] W. Yan and R. Blackwell, "Kernel Aggregated Fast Multipole Method: Efficient Summation of Laplace and Stokes Kernel Functions", 2020. arXiv:2010.15155

[8] J. Amundson, P. Spentzouris, J. Qiang and R. Ryne, "Synergia: An Accelerator Modeling Tool with 3-D Space Charge", *Journal of Computational Physics*, vol. 211, no. 1, pp. 229–248, 2006.

[9] P. Spentzouris and J. Amundson, "Simulation of the Fermilab Booster using Synergia", *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 215, 2015. doi:10.1088/1742-6596/16/1/029

[10] A. Friedman, D.P. Grote, and H. Irving, "Three-Dimensional Particle Simulation of Heavy-Ion Fusion Beams", *Phys. Fluids B*, vol. 4, pp. 2203–2210, 1992. doi:10.1063/1.860024

[11] R. Hockney and J. Eastwood, *Computer Simulation Using Particles*, CRC Press, 1988.

[12] J. Qiang, R.D. Ryne, Y.S. Habib and V. Decykz, "An Object-Oriented Parallel Particle-In-Cell Code for Beam Dynamics Simulation in Linear Accelerators", *Journal of Computational Physics*, vol. 163, no. 2, pp. 434–451, 2000. doi:10.1006/jcph.2000.6570

[13] P. Gibbon and G. Sutmann, "Long-Range Interactions in Many-Particle Simulation, Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms", in *Lecture Notes, J. Grotendorst, D. Marx, A. Muramatsu (Eds.), John von Neumann Institute for Computing, Jülich, NIC Series*, pp. 467–506, 2002.

[14] J. Qiang, "Fast 3D Poisson Solvers in Elliptical Conducting Pipe for Space-Charge Simulation", *Phys. Rev. Accel. Beams*, vol. 22, no. 10, p. 104601, 2019. doi:10.1103/PhysRevAccelBeams.22.104601

[15] W. Yan and M. Shelley, "Flexibly Imposing Periodicity in Kernel Independent FMM: A Multipole-to-Local Operator Approach", *Journal of Computational Physics*, vol. 355, no. 15, pp. 214–232, 2018. doi:10.1016/j.jcp.2017.11.012

[16] L. Ying, "An Efficient and High-Order Accurate Boundary Integral Solver for the Stokes Equations in Three Dimensional Complex Geometries", Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, 2004.

[17] A-K Tornberg and L. Greengard, "A Fast Multipole Method for the Three-Dimensional Stokes Equations", *Journal of Computational Physics*, vol. 227, no. 3, pp. 1613–1619, 2008. doi:10.1016/j.jcp.2007.06.029

[18] H. Zhang and M. Berz, "The Fast Multipole Method in the Differential Algebra Framework", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment (The Eighth International Conference on Charged Particle Optics)*, vol. 645, no. 1, pp. 338–344, 2011.

[19] L. Ying, G. Biros, D. Zorin and M.H. Langston, "A New Parallel Kernel-Independent Fast Multipole Method", in *Proceedings of ACM/IEEE SC'03*, 2003. doi:10.1145/1048935.1050165

[20] M.H. Langston, M.M. Baskaran, B. Meister, N. Vasilache and R. Lethin, "Re-Introduction of Communication-Avoiding FMM-Accelerated FFTs with GPU Acceleration", in *IEEE Conference on High Performance Extreme Computing (HPEC'13)*, 2013. doi:10.1109/HPEC.2013.6670352

[21] R. Kress: *Linear Integral Equations*, Springer, 1989, p. 82.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV049

STANDARDIZING A PYTHON DEVELOPMENT ENVIRONMENT FOR LARGE CONTROLS SYSTEMS *

S. Clark, P. Dyer, S. Nemesure, BNL, Upton, NY 11973, U.S.A.

Abstract

Python provides broad design freedom to programmers and a low barrier of entry for new software developers. These aspects have proven that unless standardized, a Python codebase will tend to diverge from a common style and architecture, becoming unmaintainable across the scope of a large controls system. Mitigating these effects requires a set of tools, standards, and procedures developed to assert boundaries on certain aspects of Python development — namely project organization, version management, and deployment procedures. Common tools like Git, GitLab, and virtual environments form a basis for development, with in-house utilities presenting their capabilities in a clear, developer-focused way. This paper describes the necessary constraints needed for development and deployment of large-scale Python applications, the function of the tools which comprise the development environment, and how these tools are leveraged to create simple and effective procedures to guide development.

GOALS

Python has grown in popularity since its 1991 inception, with wide use in scientific and analytic applications. The myriad libraries released to simplify complex tasks — such as Numpy and SciPy for scientific calculations, and the PyQt5 user interface toolkit for application development — have driven increased adoption within the Collider-Accelerator Department (C-AD) Controls Group at BNL. As this adoption began, many developers created simple scripts scattered across the filesystem, which grew into operation-critical applications over many years. Long-term maintenance of these scripts is difficult for future developers who must now not only learn the codebase, but also the unique project structure and procedures of dozens of disparate programs. The primary goal of the Python development environment is to alleviate the manageability issues described above.

PYTHON DISTRIBUTION

An essential requirement for Python usage in a large complex must be to establish a common base across all workstations. The Anaconda Python distribution provides exactly this, with each version containing a specific set of Python binaries and packages. Additionally, Anaconda is well-supported with first- and third-party tools to ease maintenance and deployment of distribution upgrades.

Initially, the Anaconda distribution was installed to a network mount which was globally available to all hosts, and any necessary packages (in addition to the standard set) were installed directly upon request. However, two primary issues were encountered with this method: performance and maintainability. Performance issues appeared shortly after the adoption, which manifested in slow application launch times. Latency and bandwidth limitations over the network were discovered as the probable cause. Maintainability was further impacted by cascades of package upgrades triggered by new package installations, leading to previously-working scripts and programs breaking due to backwards incompatibilities unless careful intervention was taken during the process.

The deployment philosophy for the Anaconda distribution was changed to account for both issues. As mentioned above, inhibited performance was found to stem from network latency. During investigation, locally-installed copies of the Anaconda distribution showed a marked decrease in application launch time. Resulting from the discovery, the Anaconda distribution was moved from the network share to local disk on each host requiring Python. The tool `conda-pack` [1] facilitates this by allowing system administrators to create a portable clone of the Anaconda distribution which then can be extracted and installed locally on machines for use.

On a yearly schedule, the Anaconda distribution is rebuilt and redistributed. This provides an opportunity for Python and package upgrades, base package additions, as well as other general maintenance of the distribution. The newest distribution is constructed on a machine, tested for general compatibility, and finally deployed to all machines using the `conda-pack` tool as described above. Two local copies of Anaconda are maintained upon release of a new distribution — the new version and the prior version – guaranteeing Applications two years of first-class support, allowing developers to migrate to newer Anaconda releases at their own pace. Distributions prior to the two locally-kept copies are available on the network share. This limits the local disk usage by Anaconda versions, but reintroduced issues with network latency. This, however, should be mitigated if applications are regularly maintained and released using the latest available distribution.

PROJECT CREATION & ORGANIZATION

A unified project architecture is key for ongoing maintainability and standardization of development procedures. The details of how projects are structured at the file level are equally important as aspects of development such as a standard package base, ensuring that developed procedures work equally across any project created within the development environment.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A custom utility, `caddy`, was created to ensure all projects follow a unified layout. Caddy is based on the open source Python package CookieCutter, which allows for the development of project templates consisting of boilerplate files and code for a project. The templates utilize a syntax which allows developers to insert values into the boilerplate based on input received when creating the initial project; these values, which may include the project name, author, and options to Git-version the project, modify both file names and contents to suit the new project. In addition to the basic project structure and contents, the templates may also optionally include a script to be run when a project is initially created. The development at C-AD heavily relies on Python virtual environments, so this post-creation script automatically spins up a virtual environment and installs base requirements upon project creation, as well as sets up a Git repository for the new project. Templates have been defined for several types of project bases — such as graphical user interfaces, command line applications, web applications — and placed into Git repositories for later reuse.

The templates used by Caddy were designed to conform to the Python Packaging User Guide (PyPUG). Following PyPUG allows all applications and packages to be installed into virtual environments; this simplifies development and deployment as is discussed later, and prevents potential lock-in that a nonstandard packaging system would introduce. Additionally, in-house packages which may benefit the broader community can easily be shared on package hosts like the Python Package Index.

The process of creating a project from a template has been automated by the Caddy script to ease barriers to entry and to minimize user error. The creation process consists of two parts: selecting a template, and defining values. The user is given a list of available templates obtained from GitLab, and makes a numerical selection for the desired project type. Then Caddy prompts the user for each variable used within the template and ensures valid values are given for each. Once complete, Caddy clones the template into a new directory specified at the prior prompt, performs template variable substitution, and runs the post-generation script associated with the template. At this point, the user is informed of the new project location and may begin development. The automation afforded by setting up projects through Caddy allows developers to focus less on project setup and boilerplate, and more on the business logic of the application. Refer to Fig. 1 for a demonstration of this process.

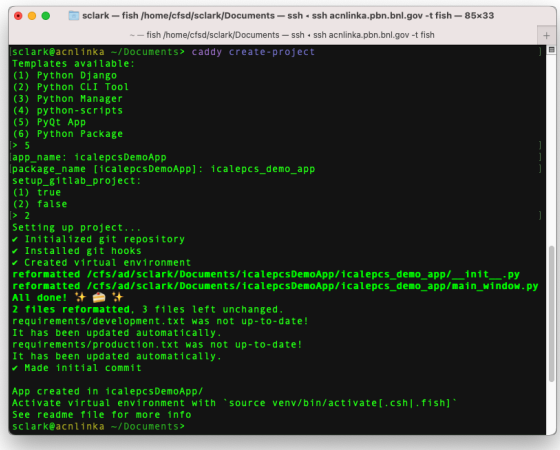


Figure 1: Use of `caddy` to create project, include template selection & variable definition.

VERSION MANAGEMENT

Version management is another critical piece of Python development given the speed at which programs evolve. When compared to other version control systems (VCS) such as ClearCase, Git provides three main benefits: free software, mature and complete implementations, and broad community support. These aspects make Git a modern, well-known, and cost-effective choice for development. Git offers a relatively simple set of core features which can be expanded upon over time, easing the learning curve for those new to Git.

A platform to host source code and version information is also required. Many solutions exist for Git-based repositories, and GitLab was chosen out of the offerings due to the availability of a free, self-hosted version of the platform, as well as additional project management features included in the platform. The self-hosted option offers greater control of the installation and data storage, while still allowing flexibility to migrate to an enterprise tier with additional features and commercial support. GitLab's primary feature, aside from version control, is a robust continuous integration and deployment (CI/CD) system. At C-AD, this has been leveraged to automate the building and deployment of Python packages and applications; this is further discussed below.

From a conceptual standpoint, a coherent version numbering system is necessary to facilitate tracking and upgrading various packages and applications within the development environment. The Semantic Versioning Specification (SemVer) was used, which operates under the principle that version numbers contain intrinsic meaning by defining major, minor, and patch revisions. For example, a package version `v3.21.5` is known to be the 3rd major revision, 21st minor revision, and 5th patch revision. When used as defined in the specification, a change in the major revision indicates that the new release breaks backwards compatibility. Changes in minor or patch revisions must maintain backwards compatibility however, and either add new features or fix existing bugs, respectively. This standard allows developers to be aware of changes they may encounter when upgrading dependencies for a project, and also ensures the smooth installation of packages into virtual environments as is defined by the PEP440 specification used by Python. [2]

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV049

DEVELOPMENT & DISTRIBUTION

Applications

Applications created in the development environment must explicitly disallow untracked modification of released executables. Releasing Python source files directly to a writeable directory was initially viable; however, as programs increased in complexity, multi-file applications became unmanageable and prone to untracked changes. The following requirements for a Python application build system were identified: performance must be responsive for user-facing applications; all necessary files and any package dependencies must be bundled with the application; and changes to released applications must be made through version control to ensure all version history is tracked.

Various solutions were considered to bundle applications, including Python's native ZipApp module, Facebook's XAR technology, Pants build system's PEX, and LinkedIn's Shiv tool. ZipApp was ruled out as its limitations on inclusion of C-extension module created a cumbersome caveat for developers. [3] XAR showed initial promise in its ability to bundle all source files and dependencies, but was found inadequate due to the overhead of cutting-edge, OS-level dependencies which are unsupported by less recent Linux distributions. A significant trial was performed with PEX, but despite having all features necessary, launching of bundled executables was not performant enough for user-facing tasks due to reliance on the `pkg_resources` library. [4] Finally, Shiv was found marketed as a performance-optimized tool with all features offered by Pex. After many weeks of testing, Shiv was found to satisfy all feature and performance criteria.

Shiv allows applications to be released as pseudo-binary files, whereby all source code and dependencies are bundled into a ZIP file containing a specific syntax to allow execution by the Python interpreter. Since all contents are contained within a single archive, modification of released applications is more difficult than helpful; one would need to extract the archive, make changes, re-archive with the correct Python-specific syntax, and copy over the existing binary. This obfuscation enforces developers' use of version control for projects.

Further, since all applications bundle dependencies within the binary, different applications may rely on differing package versions. This gives developers additional flexibility to upgrade dependencies (or not), irrespective to the underlying Python distribution. Some larger dependencies, such as PyQt5, would greatly increase disk usage by executables if they were to be included in each archive, so often-used packages are included as a part of the Anaconda distribution so any application can access those packages without bundling them in the archive.

Packages

Initial Python modules developed were released directly as source into a common directory on the `PYTHONPATH`, making the modules accessible to Python scripts system wide. This mode of development was easy, as no special setup was needed to use them. But, as some breaking changes were introduced to these common modules, any dependent script or application broke immediately with no procedure to revert to a prior version. So a method to version custom packages — just as is done through the Python Package Index (PyPI) — was necessary.

The development environment, as mentioned previously, specifies that packages must be structured to conform to the Python Packaging User Guide. An effect of this is that packages are able to be built by the `setuptools` module into `tar.gz` archives for later installation. Using this capability, in-house packages are bundled into versioned archives when released; these act as snapshots of the package at release time. These bundles are then placed into a common directory and can be installed into other projects indefinitely, regardless of any future development or releases of the package. Pip, the Python package manager, was configured globally to search this common directory for packages when a user makes an installation request. Developers wishing to use first-party packages can install them into a project's virtual environment just as they would with any third-party package available through PyPI in order to enforce strict versioning, as opposed to making them available as a part of the Anaconda distribution.

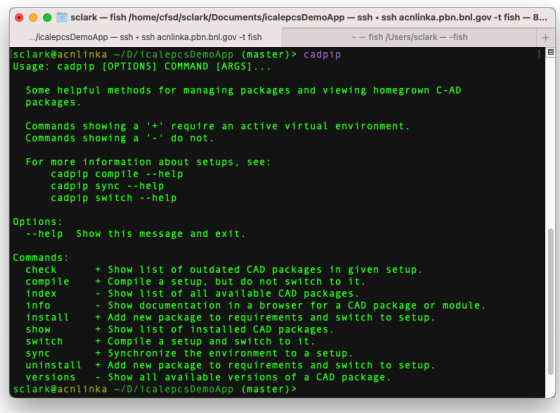


Figure 2: Various `cadpip` options for managing project packages.

Python's default Pip provides a very simple package management interface, but lacks any higher-level functions such as version conflict resolution or the notion of multiple package sets, such as for development and release. Various tools were considered for this task, such as Poetry and Pipenv, but were either immature or lacking in recent development at the time of investigation. So, a custom script `cadpip` was created to handle version resolution and package set management. Package requirements are defined in two files: `requirements/production.in` and `requirements/development.in`. Packages listed here can either be loose or pinned, meaning any version may be used or a specific version is required, respectively. Running the `cadpip switch {p,d}` command updates pack-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ages to the latest available versions which satisfy the constraints in the `production.in` and `development.in` files, respectively. It then uses `piptools` to bring the current virtual environment into sync with the selected package by adding, updating, and removing packages as necessary. The final set of package versions which were installed are then written to the files `requirements/production.txt` and `requirements/development.txt` depending on which package set a user chose; these files always contained pinned packages, and are used to exactly recreate the virtual environment when a project is either cloned or built. Refer to Fig. 2 for a sample of the tool's capabilities.

RELEASING PROJECTS

GitLab's CI/CD feature is leveraged during the release process, as mentioned in the Version Control section, which hooks into Git procedures to trigger predefined actions on a remote server. The actions performed when a project is released include: optional unit testing with PyTest, building projects using either Shiv or setuptools (described above), and releasing the build artifacts to appropriate locations for use.

The benefits of using CI/CD as opposed to building and releasing locally are two-fold. First, projects being built are guaranteed to be in an unmodified environment. If a user were to build a project locally, modifications to the Python environment may impact the end product, such as packages installed in the home directory or changes to the `PYTHONPATH` environment variable. CI/CD ensures that the build environment is unchanged by running builds in a sandboxed virtual environment under a dedicated account which maintains the default settings. So, if tests pass and a package builds, it can reasonably be expected that the application or package will work on any machine setup with the Anaconda distribution used for the build. Additionally, by executing all build and release procedures from a dedicated account, the paths applications and packages are released to may remain read-only to other users to prevent any modifications.

Bringing together semantic versioning with the CI/CD process is a lightweight shell script, `git release` (as shown in Fig. 3). `git release` makes the release process as simple as a prompt for users. It parses the Git history to find the most recent release number, and calculates new version numbers for major, minor, and patch releases. The user may select one of the three, which `git release` then tags the most recent commit with and pushes to GitLab. This formally kicks off the release process, at which point `git release` displays a progress bar tracking the build process from GitLab. If any errors occur, they are also printed to the console for inspection.

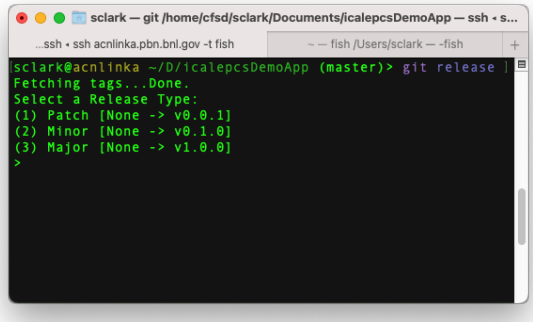


Figure 3: `git release` prompting for version selection prior to releasing software.

REFERENCES

[1] J. Crist. "Conda-pack." (2017), `https://conda.github.io/conda-pack/`

[2] N. Coghlan and D. Stufft, *Pep 440 – version identification and dependency specification*, 2013.

[3] P. S. Foundation. "Zipapp — manage executable python zip archives." (2021), `https://docs.python.org/3/library/zipapp.html#caveats`

[4] LinkedIn. "Shiv: Motivations & comparisons." (2021), `https://shiv.readthedocs.io/en/latest/history.html`

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV050

DevOps AND CI/CD FOR WinCC OPEN ARCHITECTURE APPLICATIONS AND FRAMEWORKS

R. P. I. Silvola, CERN, Geneva, Switzerland
L. Sargsyan, A. Alikhanyan National Laboratory (former YerPhI), Yerevan, Armenia

Abstract

This paper presents the Continuous Integration and Continuous Deployment (CI/CD) tool chain for WinCC Open Architecture applications and frameworks developed at CERN, enabling a DevOps oriented approach of working. By identifying common patterns and time consuming procedures, and by agreeing on standard repository structures, naming conventions and tooling, we have gained a turnkey solution which automates the compilation of binaries and generation of documentation, thus guaranteeing they are up to date and match the source code in the repository. The pipelines generate deployment-ready software releases, which pass through both static code analysis and unit tests before automatically being deployed to short and long-term repositories.

The tool chain leverages industry standard technologies, such as GitLab, Docker and Nexus. The technologies chosen for the tool chain are well understood and have a long, solid track record, reducing the effort in maintenance and potential long term risk. The setup has reduced the expert time needed for testing and releases, while improving the release quality.

INTRODUCTION

The CERN Industrial Control Systems Group of the Beams Department (CERN BE/ICS) [1] provides support and software solutions to the SIMATIC WinCC Open Architecture (WinCC OA) [2] community for setting up SCADA applications. This includes general infrastructure (cryogenics, electricity, radiation protection, etc) as well as controls for the experiments, and associated institutes. The group provides a CERN specific distribution of WinCC OA, repackaging the software released by ETM Professional Control, a Siemens AG subsidiary. In addition, the group develops and maintains WinCC OA applications, as well as frameworks for building such applications, and for connecting them to the CERN IT infrastructure.

The WinCC OA software catalogue supported by the group spans hundreds of applications and two frameworks composed of a large set of components, totalling up to millions of lines of source code written in multiple languages, including C++, PL/SQL, CTRL – which is a WinCC OA proprietary scripting language, and others. Many of these projects are required to run on both Linux and Windows, increasing their complexity.

While the code itself has always been kept in a central repository, the procedures for building a release, including compilation, packaging and deployment to our package repositories, were left to the developer in charge of each individual project. This approach meant that each component was built in a different way, and releases were only possible to be done when the expert was present. Much of this was standardized by the introduction of ARES [3], yet the approach taken was in a way upside down, requiring many manual steps for releases, leading to automated commits to the repository. This required developers to change contexts each time a release was to be prepared, be it for validation or for production, and the automated commits resulted in unnecessary pollution of the commit log, making it hard to read.

While ARES greatly simplified and automated releases, there was a disconnect between development and releases. The tools used for the release infrastructure were based on technologies poorly understood by framework developers, which lead to recurrent delays while debugging issues as the ARES expert intervention was required. The tooling, while advanced, was designed with Java projects in mind, which differ radically from WinCC OA and Frameworks development.

To alleviate these issues, a deep look was taken into the development processes, and based on it, a new set of tooling was designed and put in place. The resulting DevOps processes and CI/CD infrastructure are described in the next sections.

WORKFLOW

Deciding on a workflow (see Fig. 1) for development with git, the industry "standard" git flow was adopted, later morphing closer to the GitHub flow [4]. For any given project the master branch is considered stable, yet not necessarily production quality. There is no development branch, simply bug fix and feature branches, all of which tend to be short-lived. The frequent merges promote targeted and granular development, reducing the occurrence and probability of complicated and time-consuming merge conflicts.

To start development for a bug fix or a feature, the developer manually creates a ticket in the ticketing system. The name of that ticket is then used as the branch name. On that branch each commit produces a release that can be passed on for validation by users. Once a merge request is created, unit tests are automatically run, and each subsequent push will trigger a new set of tests to be executed. After a review and with the tests successful, the branch is merged into master, resulting in a snapshot build. A tag produces a release that is automatically deployed into an array of repositories. All the steps of the process can be performed either from the command line interface, or from a single browser tab.

For a developer this is a natural workflow, while GitLab [5] allows for less technically minded people to create releases as well, after a short training session.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV050

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

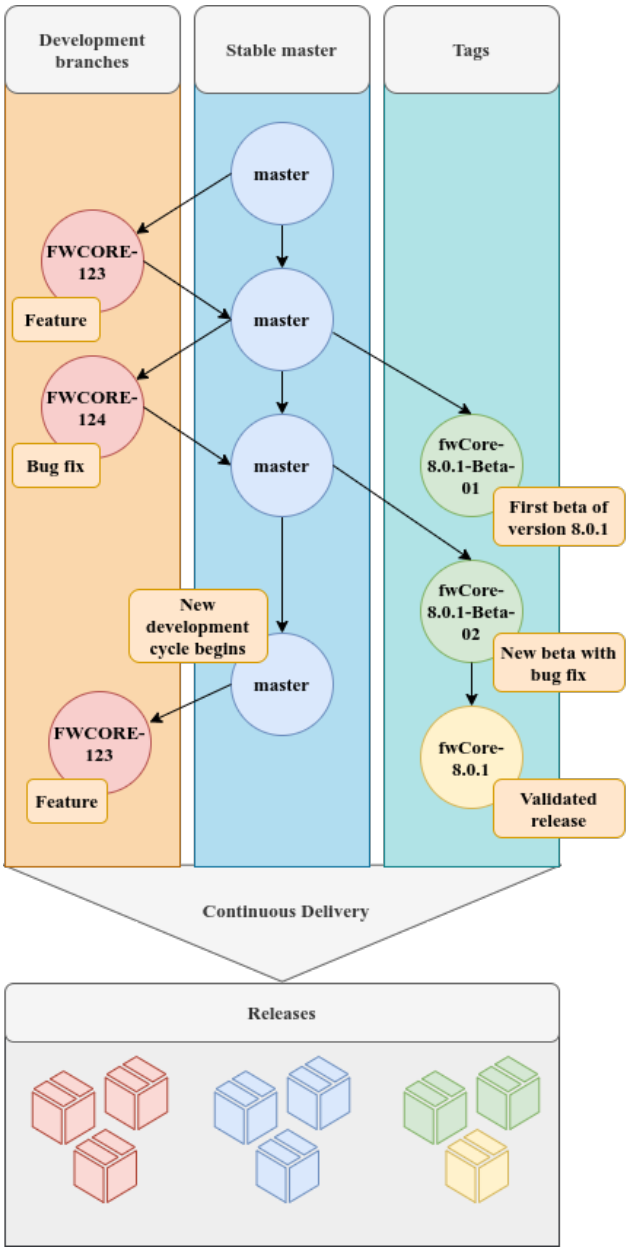


Figure 1: Example of the workflow.

RELEASE GENERATION

RELEASE GENERATION The automation of the processes is achieved by use of GitLab Pipelines, executing jobs on private hosts with Docker [6] executors, which run custom Linux images loaded with WinCC OA and a set of custom tooling. For certain quality assurance tasks other images are used to leverage newer tools. These images themselves are produced also with GitLab on the very same runners, as explained below.

CERN WinCC Open Architecture

The CERN WinCC OA Service repackages WinCC OA for Windows and Linux, in order to add in certain CERN-wide customizations – including in house license fulfilment and Oracle Database integration. This repackaged distribution is provided to the CERN WinCC OA community. As a future part of critical infrastructure, each release must be thoroughly validated to meet the highest QA standards. For this, the software package is installed on a clean production-like system, and specific common operations are executed, including creating applications and running processes. The validation has been completely automated for Linux with the use of containers, and as a final product of each automated release pipeline, a set of development and test container images are released. These are then used to compile executables developed at CERN, build components, and test further integrations.

CERN WinCC OA Framework Components

The compilation and packaging process of WinCC OA Framework components often requires careful setting up of environments, making it close to impossible without the component expert present. This problem was furthermore difficult to tackle since in the past the binaries were all stored in the version control system.

As a part of the migration to git and GitLab, all binaries were stripped from the repositories and their compilation was shifted to the release pipeline, utilizing the standard development container image. Any libraries and other build time dependencies thus must be pulled in automatically during the build, removing dependency on a single developer's environment, or their knowledge on how to recreate it.

Building the binaries from source just prior to packaging ensures they match the source code in the repository, and that no local changes make it to any component or framework release. Each release is tagged with the git revision hash, and the source code is publicly available.

After compilation, the release pipeline (see Fig. 2) passes on to the build stage, where the tooling generates documentation and manuals, updates the component XML specification used for installing them, and packages it all into a zip file. Passing into the test stage, several QA jobs are run to find issues earlier in the development cycle, producing a stable and well tested set of components for deployment.

Kobe

Kobe is the tool built to automate framework and component releases. It was originally envisioned as a leaner version of ARES, allowing for continuous builds either on GitLab or locally on the developer's system. It is built on common, historically stable and commonly used GNU/Linux tools and bases its logic on information available in the repository and in the pipeline. It has since its conception grown to do much more, including generation of Online Qt documentation integrated into WinCC OA, as well as rich PDF manuals, using LaTeX templates, and Markdown.

Kobe produces fully framework compatible components with a guaranteed unique version number for each branch, tag and revision. Releases follow a specific format with the component or framework name, the version number and an optional beta or release candidate suffix. The lack of either

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV050

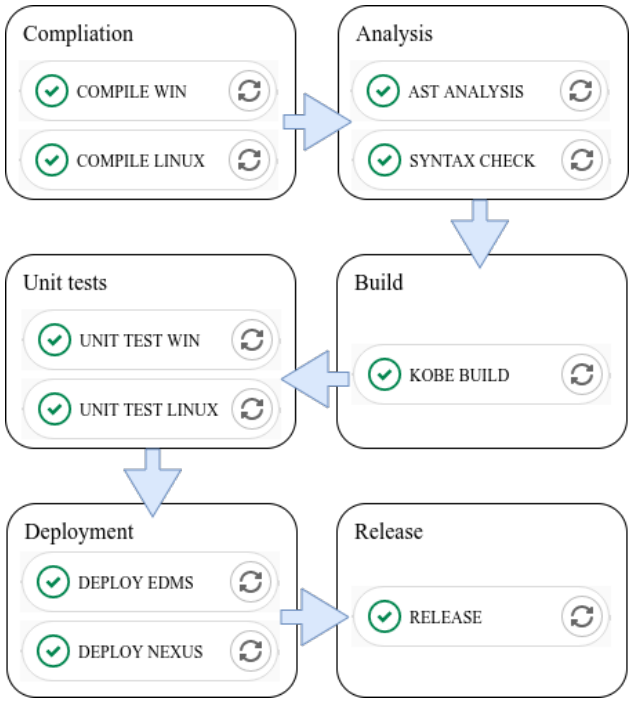


Figure 2: Example of a release pipeline.

suffix indicates a final release of that version, leading to the packages of components from these pipelines getting labelled accordingly, and to be released in to the appropriate repositories – after having passed through automated tests. These unique versions and suffixes allow dependable identification of deployed versions, thus reducing the ambiguity what has been deployed.

CERN WinCC OA Frameworks

Frameworks are built in very much the same way, simply as a collection of components. The collections are specified with a configuration file and Kobe automatically pulls the the components from the appropriate repositories. This could be from a GitLab job, a Nexus [7] repository or from EDMS [8].

During active development, the latest snapshot builds are included into the framework snapshot distributions, which are triggered at the end of each component snapshot pipeline. To ensure the possibility to generate new versions of the full framework at any time in the future, the versions included for framework beta and final releases are locked and pulled from mid- or long-term repositories.

QUALITY ASSURANCE

Quality assurance is an integral part of the CI/CD pipeline, and includes both static analysis as well as unit tests. For C++, abstract syntax trees are analysed against thousands of carefully selected rules from publicly available rule sets, including MISRA [9], HIC++ [10] and Fuchsia [11], among others, and a report is produced and made available in a centralized reporting tool. This allows us to identify hidden bugs and vulnerabilities, as well as code duplication and smells, all with historical trends. As abstract syntax trees are only possible with compiled languages, a different approach is taken for CTRL by analysing it with WinCC OA's own syntax checker. The results are transformed into standards compliant reports, visible with details directly in the GitLab Pipeline UI. In the future it is planned to have these also integrated into centralized reporting tool.

Unit tests implemented according to specifications drawn up together with the component responsible and users' feedback are included in weekly containerized component tests, daily full stack integration tests on physical machines as well as triggered full framework tests of the JCOP framework [12] and the Unified Industrial Control System and Continuous Process Control framework (UNICOS-CPC) [13]. A CERN custom interface to WinCC OA and a framework unit test component are used to automate the tests. A clean application is instantiated in a container, with the set of components to be tested. The test results generated by the unit test component are likewise in standards compliant format and thus visualized directly on GitLab Pipeline UI. They are also fed into the centralized reporting tool, improving their visibility.

DEPLOYMENT

Once packaged and tested, components and frameworks are deployed to repositories. Each repository serves its own purpose, and depending on the type of release, it is routed to a different one, completely automatically.

GitLab

For active development, GitLab job artifacts act as the short-term repository. With GitLab API [14] they can be pulled programmatically either to a project, or to build a framework. Based on Amazon S3 [14] compatible object storage, the releases are quickly available, and are stored generally for a week. The artefacts of the latest job are stored indefinitely, as are the artefacts of tags – i.e. betas, release candidates and final releases.

Nexus

All betas, release candidates and final releases are automatically deployed to Sonatype Nexus repositories. We do not take advantage of staging repositories, as that purpose is served by GitLab's object storage. Nexus repository is a mid-term repository and is available in parts of the CERN network, but not outside of it, nor is it available in the experiment networks. Thus, it is mainly used for validation and pre-production tests.

EDMS

Final releases are deployed to EDMS, our most long term document repository (more than 30 years). It is also the most accessible one, available in all CERN networks – experiments included – and outside of CERN. The ARES EDMS client is used for the deployments to EDMS, as it greatly simplifies the tasks needed to follow EDMS release workflows. Improvements to the tool have been made to allow

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV050

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

better diagnostics of deployments and the document statuses, increasing its robustness.

NFS

According to previously mentioned requirements we will now provide details about hardware used and specifications.

INFRASTRUCTURE

GitLab, provided as a service by CERN IT Department, is used as the backbone of the CI/CD infrastructure as illustrated on Fig. 3. The integrated container registry is used to store container images, and the UI is used to configure all of it. CERN GitLab Service also provides public runners, which are sufficient for most jobs. WinCC OA development and testing however have very specific requirements, necessitating the use of private runners.

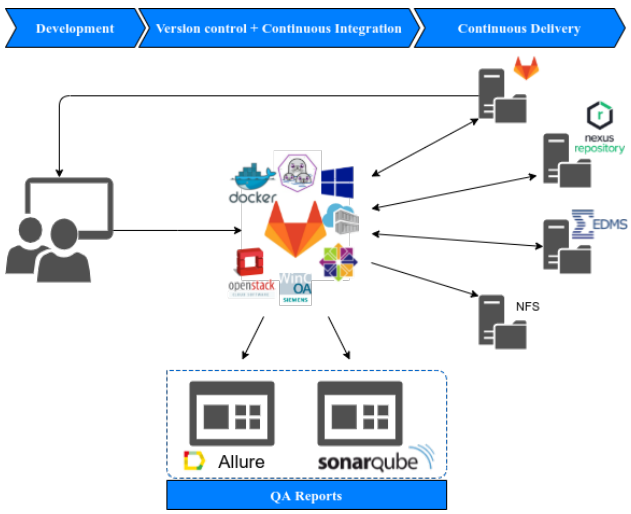


Figure 3: CI/CD infrastructure.

For lightweight jobs, OpenStack [16] Linux and Windows virtual machines are used. Setting up private runners on them guarantees availability for our frequent jobs as well as optimizes the pulling of containers – which are fairly large when loaded with WinCC OA. This has also allowed exploratory work towards alternative container based custom executors which will be necessary for the future operating systems.

For heavier tasks bare metal servers are necessary. Repackaging of WinCC OA distributions for Windows and Linux are I/O heavy processes that take over an hour on virtual machines, yet happen in just minutes on dedicated hosts. Our full stack integration tests benefits as well from their dedicated bare metal runner, as the tests require a full production-like system with connected devices, including PLCs, to run tests on multiple levels, from low-level front-ends to middleware and SCADA integration.

Exposing a bare metal system with a shell runner to a common GitLab instance is a compromise in security, and great care is taken to strictly restrict their access.

CONCLUSION

Approximately twenty-five thousand framework snap-shot releases, triggered by a matching twenty-five thousand component snapshot releases, and hundreds of thousands of more development releases have been produced since we started deploying Kobe build infrastructure in the late summer of 2018; not counting hundreds of betas, release candidates and production releases of component and frameworks. All of these releases have passed automatic tests in containers running the latest releases of WinCC OA, which themselves were generated and tested automatically by our CI/CD tooling.

This is an increase of two orders of magnitude from the old automatic release service, while reducing the effort for the developers to close to zero, saving thousands of hours of work only in releases, not to mention the increase in quality of the software. What used to be an error prone and complicated procedure involving multiple different web applications, now takes all of writing ‘git tag‘ and pushing the tag to GitLab. For snapshot and development builds, pushing the sources to GitLab is all there is to it, making it a natural part of development.

The developers thus benefit from a more natural, development-oriented workflow, while the users gain more frequent releases, shorter lead times and simply a better tested end product, all with publicly available reports and up to date documentation.

The quality of support has also improved as we can guarantee version uniqueness, as well as provide instant test releases for any bug fixes, to be validated in production-like environments, while being clearly tagged as development releases. Trusted users can even create merge requests, which will immediately provide them with a usable release of the component.

The added control and improved quality assurance of the strict and replicable release flow also improve the operational safety and security on the organizational level, while further security tooling can now be easily integrated.

With conventions and templates, we have arrived at a highly standardized and unified system. This is not to say everything is the same, and that there are no exceptions.

Most compilations are configured with qmake, on Windows and Linux, yet some developers have preferred to stay with pure make or CMake builds. This knowledge of build system is however abstracted away by scripts and templated pipelines, and while debugging a failing compilation might require expert knowledge, most development can be done by junior developers with limited experience.

Further customizations are also made available through environment variables, allowing for custom repository structures and non-standard naming conventions, among other things. While we approach near complete unification of our conventions in the group, as the tool is being adopted by experiments and application groups around CERN, legacy structures and conventions need to be supported, and as with any software, further improvements will al-ways be needed.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV050

The success of the project hinged on the continuous involvement of the entire team. Their contributions ensured the fit for our needs, and what was seen was a true integration of new and existing tooling and processes. The team's involvement also ensured the project's adoption, and its ever-continuing improvement.

REFERENCES

[1] CERN BE/ICS, https://be-dep-ics.web.cern.ch/

[2] SIMATIC WinCC Open Architecture, https://winccoa.com/

[3] I. Prieto Barreiro and F. Varela, "ARES: Automatic Release Service", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 503-507. doi:10.18429/JACoW-ICALEPCS2017-TUPHA049

[4] GitHub Flow, https://guides.github.com/introduction/flow/

[5] GitLab, https://about.gitlab.com/

[6] Docker, https://www.docker.com/

[7] Sonatype Nexus, http://www.sonatype.org/nexus/

[8] Engineering & Equipment Data Management Service, https://edms.cern.ch/

[9] MISRA, https://www.misra.org.uk/misra-c-plus-plus/

[10] HIC++, https://www.perforce.com/blog/qac/high-integrity-cpp-hicpp

[11] Fuchsia, https://fuchsia.dev/fuchsia-src/development/languages/c-cpp/cpp-style

[12] JCOP Framework, https://jcop.web.cern.ch/

[13] UNICOS Framework https://unicos.web.cern.ch/

[14] GitLab API, https://docs.gitlab.com/ee/api/

[15] Amazon S3, https://docs.aws.amazon.com/s3/

[16] OpenStack, https://www.openstack.org/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL02

DEVELOPMENT OF A SINGLE CAVITY REGULATION BASED ON MicroTCA.4 FOR SAPS-TP

W. Long†, X. Li, Y. Liu, S. H. Liu
Institute of High Energy Physics Dongguan Campus, Chinese Academy of Sciences, Dongguan, China

Abstract

A domestic hardware platform based on MTCA.4 is developed for a single cavity regulation in Southern Advanced Photon Source Test Platform (SAPS-TP). A multi-function digital processing Advanced Mezzanine Card (AMC) works as the core function module of the whole system, it implements high speed data processing, Low-Level Radio Frequency (LLRF) control algorithms and an interlock system. Its core data processing chip is a Xilinx ZYNQ SOC, which is embedded an ARM CPU to implement EPICS IOC under embedded Linux. A down-conversion and up-conversion RTM for cavity probes sensing and high-power RF source driver can communicate with AMC module by a ZONE3 connector. A hosted tuning control FPGA Mezzanine Card (FMC) combines both the piezo controlling and step-motor controlling functions for independent external drive devices. The design of the hardware and software of the platform electronics and some test results are described in this paper. Further test and optimization is under way.

INTRODUCTION

The SAPS-TP is next to the China Spallation Neutron Source (CSNS) which is located at Dongguan City, Guangdong Province in China. It is an innovative research platform for advanced accelerator and X-ray technologies serving Southern Advanced Photon Source (SAPS) which will be built as a 4th generation light source based on diffraction limit storage ring, and the CSNS II which is the upgrade of the CSNS. It is mainly composed of a superconducting RF hall, an optical experiment hall, a low temperature hall, a high-accuracy measurement hall and a comprehensive laboratory [1].

The superconducting RF hall is about 3500 m^2. It has 2 vertical test pits and a horizon test station can be applied for assembling and test of the 324 MHz spoke cavity, 648 MHz elliptical cavity, 500 MHz elliptical cavity and 1.3 GHz elliptical cavity, etc.

The function of the single cavity regulation includes amplitude and phase stabilization controlling of cavity field, resonance controlling, continue waveform (CW) or pulse mode conversion and fast interlocking, etc.

MTCA.4 system is the new generation embedded digital hardware platform specifically for high energy physics applications. It is a modular, open standard architecture with high reliability, which is becoming more and more popular, and widely used in European X-ray Free Electron Laser (EXFEL), European Spallation Source (ESS) and Stanford Linear Accelerator Center (SLAC), and so on.

A domestic hardware platform based on MTCA.4 is developed for SAPS-TP single cavity regulation. The system architecture and the development of hardware platform based on MTCA.4 are described in this paper.

SYSTEM ARCHITECTURE

The single cavity regulation for SAPS-TP is mainly used to control to RF field and the resonance frequency of the cavity. The block diagram of the system design is shown in Fig. 1.

Dedicated RF power source (klystrons or solid-state amplifier (SSA)) is used to feed the cavity with required input power. In the RTM module, the forward and reflected power signals, the cavity pick-up signal and the reference RF signal are down-converted to intermediate frequency signals using analog mixers driven by local oscillator signal (LO).

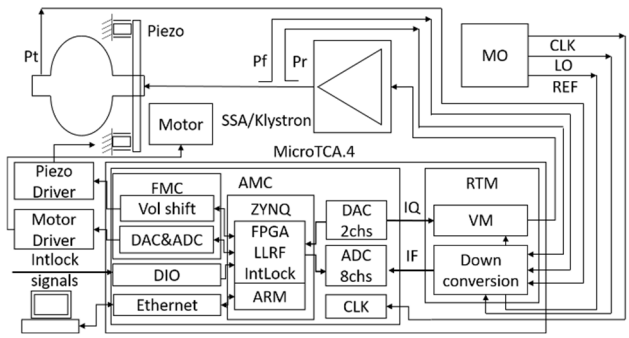


Figure 1: Design of single cavity regulation for SAPS-TP.

The LLRF system implements the cavity RF field controlling feedback loop to keep the RF field amplitude and phase stability [2, 3].

The master oscillator provides RF synchronization signals, such as the reference signal (REF), LO signal and the clock (CLK). In the AMC module, the intermediate frequency signals are sampled by ADCs with the frequency of CLK. Next, the raw data are demodulated to in-phase (I) components and quadrature (Q) components with the IQ or no-IQ algorithm. Errors between setpoint values and detected values are sent to PI controller. The digital I/Q output of PI controller are sent to DACs and are converted to analog I/Q signals. The baseband I/Q signals are then mixed with the in-phase and quadrature-phase components of a REF signal whose frequency is located at the required RF frequency to generate final RF excitation signal. This process is called up-converted which implemented by the vector modulator device of the RTM module.

The superconducting cavity is very susceptible to small changes in dimension, because of its very narrow RF resonance bandwidth. The equipped mechanical tuners can tune the cavity to a resonance frequency. There are two types of tuners, the slow tuners are based on step motors, and the fast tuners are based on piezo elements. The LLRF

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL02

system implement the cavity resonance frequency controlling feedback loop to compensate the cavity detuning caused by reasons such as Lorentz force and microphonics and so on. The digital frequency control output is composed of step motor control signals and piezo control signals. In FMC card, the digital step motor control signals from AMC module are converted to 5 V pulse signals which be sent to step motor driver, and the digital piezo control signals are converted to 0~10V analog signals by ADCs which be sent to piezo driver.

The LLRF system also implement cavity quench detection. The others interlock signals are connected to the digital IO port of the AMC module front panel such as arc, vacuum, temperature, water flow, etc. All the interlock signals are connected to the interlock system which also implemented by the FPGA of the AMC module for fast protection. The RF excitation can be turn off quickly, and the output interlock signal will switch off the RF switchers of the RTM module to cut off the excitation quickly too.

The hardware platform based on MTCA.4 is being developed. It mainly composed of the following devices :

- A domestic AMC module.
- A Module Management Controller (MMC) mezzanine Card.
- A tuning control FMC card.
- A down-conversion and up-conversion RTM module.
- A domestic MTCA.4 chassis including the two 1.6 kW power modules (one for redundancy) and 2 Cooling Unit (CU) modules.

The first version AMC module and the MMC mezzanine card have been produced and tested. The Design of the tuning control FMC card has been finished and is going to be produced. It is based on low speed and high precision DACs and ADCs, and level shift and voltage translation circuit. The RTM module has 8 down-conversion channels and 1 up-conversion channel. It has been developed and produced by the colleagues in the project group. The designed down-conversion RF frequency range is from 300 MHz to 4 GHz, IF frequency range is from 5 MHz to 100 MHz, and up-conversion RF frequency range is from 200 MHz to 4 GHz. Now, its test and optimization of the RTM module is under way. A domestic MTCA.4 chassis including two 1.6 kW power modules and 2 CU modules has been produced by our cooperation manufacturer. Its functional test is finished and the power ripple index need to be improved.

DEVELOPMENT OF AMC MODULE

We developed a domestic multifunctional digital processing AMC module which is shown in Fig. 2. It is expected to be used for vertical and horizontal tests of RF cavity in SAPS-TP, and can also meet the requirements of different systems in CSNS II and SAPS, such as RF system & Control system & Beam diagnostic system.

The AMC module has good compatibility. It is conforming to MTCA.4 standard and the zone3 IO connector assignment class A1.1 CO with double-width mid-size dimensions. The AMC module provide 4-lane PCI Express Gen3 interface connected to port 4~7 of the backplane to realize high speed data transmission with other AMC modules. It also realizes 1GbE Ethernet interfaces (Port 0~1), point-to-point link interfaces (Port 12~15), M-LVDS interfaces (Port 17~20), TCLKA, TCLKB and FCLKA which conform to standard MTCA.4 backplane topology. So, it can work very well as part of a large system based on MTCA.4. What makes it special is that we designed the AMC module and expected that all the digital functions of a single cavity regulation can be finished by one AMC module. So, there is no need to communicate through high-speed serial buses, and the delay of control system is reduced.

Figure 2: Domestic digital processing AMC module.

It has 8 channels ADCs and 2 channels DACs. The Analog Devices AD9268 ADC chip with maximum sampling rate up to 125 MHz and 16-bit resolution is chosen for IF signals sampling. Its analog input bandwidth is 650 MHz, which enable this ADC chips can also be used for 324 MHz or even up to 648 MHz direct sampling applications. A compatible analog front end is designed to adapt to different applications. All the ADC input channels can be configured to AC coupled or DC coupled which be routed to the different differential pairs (TF or PA) of the Zone3 connector conforming to DESY Zone3 class A1.1 CO assignment.

The DAC chip is Analog Devices AD9783. Its dual DA architecture facilitates easy interface to common quadrature modulators. Its sampling rate is up to 500 MHz, and has16-bit resolution. This chip is widely used in CSNS RCS RF system and its performance and stability has been tested.

Because the ZYNQ SOC IO resources is very limited, all the digital interface of AD and DA chip are DDR LVDS interface, so 2 channels can share a group of IO pins.

The core data processing chip is a Xilinx Zynq-7000 SOC system-on-chip. The ZYNQ SOC combines a dual core ARM processor, with a traditional FPGA logic fabric. The single silicon chip can be used to implement the function of an entire system rather than several different physical chips being required. In the single cavity regulation, an EPICS IOC runs in PS part of the SOC which is the ARM processor for remote control, and a LLRF algorithm and interlock system runs in PL part of the SOC which is the FPGA. The data communication between FPGA and ARM

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

processor is carried out through AXI bus inside the ZYNQ SOC chip. It simplifies the architecture of hardware greatly, and saves the PCB space which is important to AMC module, because it is not very big.

The Ethernet interface connected to ARM CPU is used for network communications such as remote control and monitoring.

The clock tree is designed to realize the synchronization of signals on board. The clock source can be chosen from the internal crystal oscillator or external input clock signal. There are three input methods for the external clock:

- The front panel clock input.
- The RTM_CLK clock signal from RTM module.
- The TCLKA/TCLKB clock signal from backplane.

The AMC module is designed with ultra-low clock jitter. We expect that the phase error will be smaller than 0.1°, if a 648 MHz direct sampling application based on it.

A 12-channel jitter attenuator SI5394 is used to perform jitter attenuation, and two LTC6950 frequency synthesizer are used for clock distribution. The total theoretical clock jitter is about 170 fs RMS, but not yet be tested due to the unfinished adapter RTM module.

An FMC connector is provided which fully compatible with the FMC Low Pin Count (LPC) standard. It is mainly designed for multi-channel step motor controlling and piezo controlling, and maximally compatibles with commercial FMC boards such as DFMC-AD16 and DFMC-MD22, etc. It is also extended with 3 additional high-speed serial ports of High Pin Count (HPC). So, the interface can also compatible with some usual 4 channel fiber communication FMC card.

The front panel digital IO on LEMO 00 connector has 10 pins and conform to 3.3 V LVTTL standard which can be used for external trigger and interlock signals.

A 4 channels QSFP cage is provided in front panel. And each channel can be used separately for timing, timestamp, interlock signals and optical fiber communication.

This AMC module has 2 GB DDR3 memory size (1 GB memory connected to ARM CPU, 1 GB memory connected to FPGA).

The block diagram of the AMC module is shown in Fig. 3.

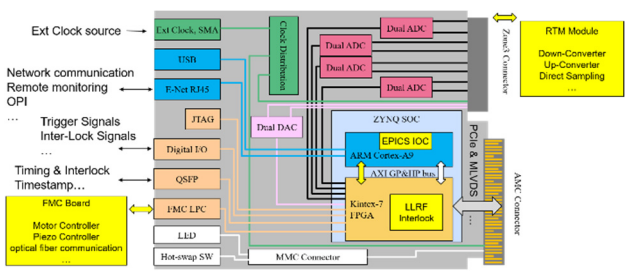


Figure 3: The block diagram of the AMC module.

Because the RTM module is being tested and optimized yet, performance of the AMC module is tested with the compatible signal conditioning input card SIS8900 RTM module and a developed adapter RTM module.

The Fast Fourier Transform (FFT) spectrum of the ADC is shown in Fig. 4.

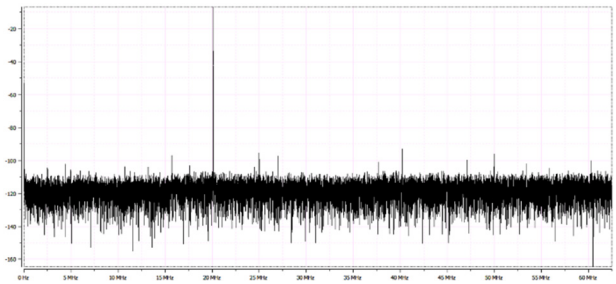


Figure 4: FFT spectrum of ADC.

The results of Signal-To-Noise Ratio (SNR) measurement are from 75.9 dBFS to 76.2 dBFS, and the results of (Spurious-Free Dynamic Range) SFDR measurement are from 86.2 dBFS to 92.1 dBFS.

To test channel isolations, we feed IF signals with different frequencies into the two adjacent channels, and the FFT spectrum of a channel is observed by spectrum analyser. The cross talk signal from the adjacent channel can be observed. The results of the isolation test is shown in Table 1.

Table 1: The Results of Isolation Test

[dB]	Ch0	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7
Ch0	-	**76.0**	-	-	-	-	-	-
Ch1	**84.3**	-	**80.9**	-	-	-	-	-
Ch2	-	**75.2**	-	**77.1**	-	-	-	-
Ch3	-	-	**82.2**	-	**80.8**	-	-	-
Ch4	-	-	-	**87.3**	-	**79.5**	-	-
Ch5	-	-	-	-	**78.9**	-	**85.6**	-
Ch6	-	-	-	-	-	**94.8**	-	**85.1**
Ch7	-	-	-	-	-	-	**78.1**	-

We also tested the performance of DAC. The result of SNR measurement is 78.1dB, but there are some bigger harmonic components. The analog circuit behind the DAC output is going to be optimized.

MMC MEZZANINE CARD

The MTCA.4 system provided a remote and centralized health management service for the modules in chassis which is based on the Intelligent Platform Management Interface (IPMI) specification. The MicroTCA management architecture is mainly realized by a MicroTCA Carrier Management Controller (MCMC) and MMC controllers which connect each AMC module to Intelligent Platform Management Bus (IPMB) based on I^2C. MMC controller is a very important part of the AMC module. Without the MMC controller, AMC module will not be detected and identified by MicroTCA system, and the payload power will never be provided for it.

We developed, based on N.A.T NAMC-MMC-Reference Design, a MMC controller mezzanine card for our AMC module. It can be connected to the AMC module by a 40 pins connector. Considering the difficulty and the development time. There are 3 phases of our development:

- Firstly, both the hardware and software are based on the reference design.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL02

- Secondly, we had developed our own MMC controller software.
- Thirdly, we'll going to design and develop our own MMC controller including all the hardware and software.

We test the MMC controller with the nVent 3U commercial MTCA.4 chassis and NAT-MCH-PHYS80 MTCA Carrier Hub (MCH). Module operational state management, hot swap management, Light-Emitting Diode (LED) management have been tested, and all these functions work very well. The AMC module can be detected and fed with payload power normally. And we also did the voltage sensors and temperature sensors reading test and the temperature threshold event test, all these functions work well too.

The design of MMC controller software is divided to three layers. At the bottom, there is the driver layer including I^2C drivers, voltage sensor drivers and temperature sensor drivers. All the values of sensors are periodically read into a pool which is a list of arrays. The middle layer implements the state machines for module status switching, hot swap management, LED management and threshold event management, etc. The top layer is in charge of IPMI communications including protocol packet parsing and building. This design of software architecture provides many conveniences for debugging and maintenance. So, for MMC testing, a fake temperature threshold event IPMI package can be easily built and send to MCH which will control CU module to increase speed of fans to decrease the AMC module temperature.

The MMC mezzanine card and its test bench are shown in Fig. 5.

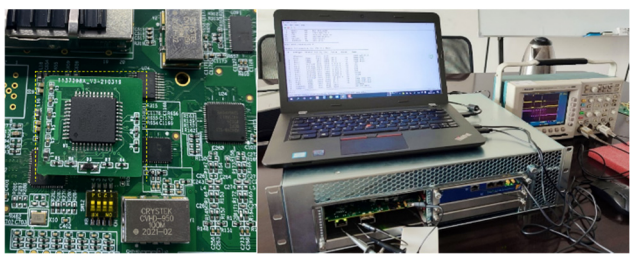


Figure 5: MMC mezzanine card and its test bench.

SOFTWARE DEVELOPMENT

An embedded Linux is developed for the Zynq7045 SOC chip based on open-source Linux kernel source and the Xilinx PetaLinux tools. The experimental physics and industrial control system (EPICS) Input/Output controller (IOC) application Demon is also developed and tested. The Demo of EPICS IOC has 3 types of Process Variables (PVs) which are used to test reading and writing of FPGA registers and reading of waveforms respectively. The data of waveform PV is transferred from FPGA via Direct Memory Access (DMA). A Xilinx AXI DMA intellectual property (IP) core manages and controls the high-speed transmission the data from FPGA to ARM. It be configured to simple mode to fetch data from a FIFO. A user defined IP is developed to control the AXI DMA timing and provide the interface between user data and the Xilinx AXI DMA. The Linux DMA driver for Xlinx AXI DMA is developed based on Xilinx DMA proxy driver which is an open source code and can be download from Xilinx web site. The device support waveform PV realizes the user space operation of DMA based on the Linux DMA kernel module. The entire system including the EPICS IOC demo is no more than 100 MB and can be loaded via a Secure Digital Memory (SD) card for convenient debugging.

SUMMARY

This paper mainly describes the system architecture of the single cavity regulation for SAPS-TP which will be build based on MTCA.4 hardware platform, the hardware and software design and development of platform electronics. The hardware platform has good compatibility and extensible. The developed domestic AMC module provides rich functions and standard extension interface. It is expected to constitute a complete single cavity regulation control system including RF field controlling, resonance frequency controlling and interlock system with the tuning control FMC card and a down-conversion and up-conversion RTM module. There is no need to communicate through high speed serial buses, the system architecture is simplified and the delay of control system is reduced.

REFERENCES

[1] S. N. Fu and S. Wang, "Operation status and upgrade of CSNS", in *Proc. 10th International Particle Accelerator Conference (IPAC2019): Melbourne, Australia, 2019*, pp. 23-27. doi:10.18429/JACoW-IPAC2019-MOZPLM1

[2] K. Przygoda *et al*., "MicroTCA. 4 based Single Cavity Regulation including Piezo Controls", in *Proc. 7th International Particle Accelerator Conf. (IPAC2016)*, Busan, Korea, 2016, pp. 3152-3154. doi:10.18429/JACoW-IPAC2016-THOAA03

[3] Q. Wang, J. Dai, *et al*., "Development of a 166.6 MHz digital LLRF system for HEPS-TF project", in *Proc. 19th International Conference on RF Superconductivity (SRF'19): Dresden, Germany. 2019*, pp. 1073-1077. doi:10.18429/JACoW-SRF2019-THP075

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

R&D STUDIES FOR THE ATLAS TILE CALORIMETER DAUGHTERBOARD * †

Eduardo Valdes Santurio‡, Samuel Silverstein, Christian Bohm,
Suhyun Lee, Katherine Dunne, Holger Motzkau
Stockholm University, Stockholm, Sweden

Abstract

The ATLAS Hadronic Calorimeter DaughterBoard interfaces the on-detector with the off-detector electronics. The DaughterBoard features two 4.6 Gbps downlinks and two pairs of 9.6 Gbps uplinks powered by four SFP+ Optical transceivers. The downlinks receive configuration commands and LHC timing to be propagated to the front-end, and the uplinks transmit continuous high-speed readout of digitized PMT samples, detector control system and monitoring data. The design minimizes single points of failure and mitigates radiation damage by means of a double-redundant scheme. To mitigate Single Event Upset rates, Xilinx Soft Error Mitigation and Triple Mode Redundancy are used. Reliability in the high speed links is achieve by adopting Cyclic Redundancy Check in the uplinks and Forward Error Correction in the downlinks. The DaughterBoard features a dedicated Single Event Latch-up protection circuitry that power-cycles the board in the case of any over-current event avoiding any possible hardware damages.

We present a summary of the studies performed to verify the reliability if the performance of the DaughterBoard revision 6, and the radiation qualification tests of the components used for the design.

INTRODUCTION

The instantaneous luminosity at HL-LHC will be increased by a factor of five compared to the LHC. Consequently, the read-out systems of the ATLAS detector [1] will be exposed to higher radiation levels and increased rates of pileup. The current electronics for the read-out system of the ATLAS Tile Calorimeter (TileCal) will not be able to handle the new requirements imposed by the HL-LHC. R&D work is ongoing to upgrade the TileCal electronics with a new design that will provide continuous digital read-out of all the calorimeter cells with better energy resolution, improved timing and less sensitivity to out-of-time pileup [2]. The upgrade R&D work requires Total Ionizing Dose (TID), Non Ionizing Energy Loss (NIEL) and Single Event Effects (SEE) tests to be performed on the upgraded on-detector electronics, to qualify the design as reliable for the HL-LHC radiation environment.

* Work supported by Stockholm University and CERN.
† Copyright 2021, CERN for the benefit of the ATLAS Collaboration. CC-BY-4.0 license.
‡ eduardo.valdes@fysik.su.se, eduardo.valdes@cern.ch

ATLAS TILE CALORIMETER

TileCal is a sampling calorimeter with plastic scintillator tiles and steel plates as active and absorber materials, respectively. TileCal is longitudinally divided into four cylindrical barrels (Fig. 1b) each comprising 64 wedge-shaped modules (Fig. 1c). Scintillator light is collected on each side of a pseudo-projective cell by wavelength shifting fibers and read out by a pair of PMTs.

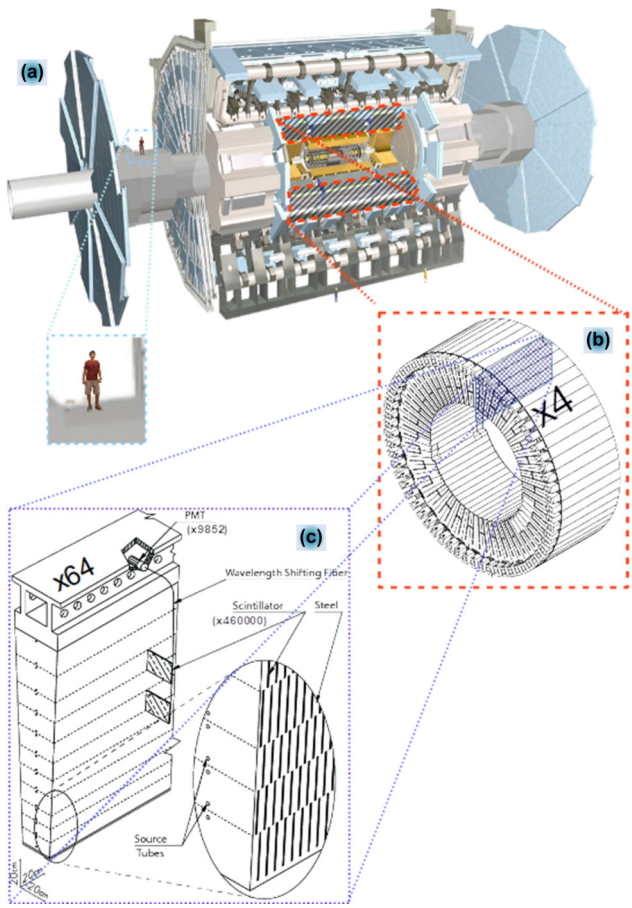


Figure 1: (a) The ATLAS detector. (b) A TileCal barrel. (c) Depiction of a TileCal wedge-shaped module.

THE TILECAL HL-LHC UGPRADE SYSTEM

The TileCal HL-LHC upgrade on-detector electronics will continuously sample data from all TileCal PMTs at 40 MHz by means of 896 independent modules, so-called MiniDrawers (MD). Each MD servicing up to six pairs of

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

PMT channels, each pair sampling the light coming from the same TileCal cell. Each PMT signal is conditioned, shaped and fed into amplifiers with low- and high-gain by a Front-end electronics (FENICS) card, before they are digitized by a MainBoard (MB, Fig. 2). A read-out link and control DaughterBoard (DB) receives and propagates LHC synchronized timing, control and configuration to the front-end, while transmitting continuous read-out of the MB channels to the off-detector systems. Each MD is independently powered by radiation tolerant Low Voltage Power Supplies (LVPS) sitting on-detector and designed to withstand the radiation requirements of the HL-LHC.

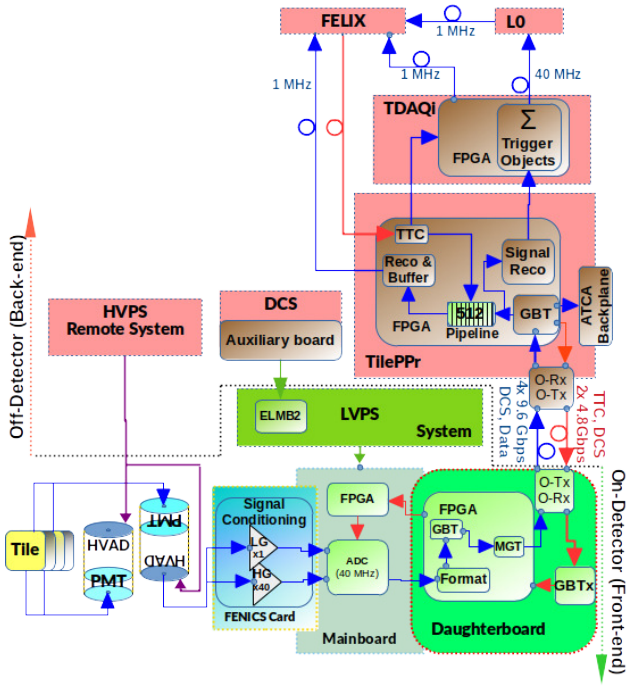


Figure 2: Block diagram for the On-Detector and Off-Detector electronics.

On the off-detector, Tile Processors (TilePPrs) receive the PMT digitized data and store it in pipelines (Fig. 2). The TilePPr will send all the digitized data to the Level 0 (L0) trigger system through the Trigger and DAQ interface (TDAQi) at 40 MHz, and the Front-End Link eXchange (FELIX) system will read-out data stored in the TilePPr pipelines at 1 MHz if a trigger decision is made. The power will be controlled, monitored and distributed to the on-detector electronics by the Detector Control System (DCS) interfaced LVPS, while the High Voltage Power Supply (HVPS) remote system will provide high voltage to the High Voltage Active Dividers (HVAD) sitting the PMTs.

The front-end electronics of TileCal include multiple, complementary test and calibration systems. These are the charge injection system (CIS), the laser calibration system and the Caesium (Cs) source calibration system. The CIS sends electronic pulses with configurable amplitude can be injected into the system in order to test the behaviour of the digitization and read-out chain of each channel over its full dynamic range. The LASER system distributes short, controlled light pulses over optical fibres to the light mixers of the PMT blocks to allow measurements of individual PMT gains of each PMT in the absence of collisions. The Cesium system uses radioactive ^{137}Cs γ sources that are moved hydraulically through water-filled tubes that pass through each scintillating tile in the calorimeter, producing strong, slow signals read by a separate read-out path is implemented with a slow integrator circuit read-out by ADCs sitting in the FENICS cards.

THE DAUGHTERBOARD REVISION 6

The DB Revision 6 (DB6, Fig. 3) design aims to solve the radiation tolerance issues found in the previous revision while incorporating a more robust timing design and extra features that will allow minimizing single points of failure one step further. The DB6 consists in a radiation tolerant and electrically redundant board that serves as a hub that interconnects the on-detector and off-detector electronics of the HL-LHC TileCal Upgrade System by means of four Small Form-Factor Pluggable+ (SFP+) modules and a 400 FPGA Mezzanine Card (FMC) connector. A block diagram for the design of the DB6 is shown in Fig. 4. The design consists in equal and electrically independent halves, each capable of servicing the six PMT channels that correspond to the half of the MD interfaced with the DB.

Two CERN radiation-hard GBTx ASICs will receive two GBT-FEC protected 4.8 Gbps downlinks with LHC synchronized clocks and configuration commands. Each GBTx recovers two 160 MHz good quality LHC synchronized clocks in order to drive the GTH transceivers of both FPGAs, two 40 MHz Trigger, Timing and Control (TTC) synchronous clocks to drive the relevant logic of the KU FPGAs, and four 40 MHz TTC phase configurable clocks to drive the digitizing blocks and calibration signals on each MD quadrant. Two Xilinx KU XCKU035 FPGAs powered by TMR capable firmware provide continuous GBT-CRC protected words to the four 9.6 Gbps uplinks with all the Detector Control System (DCS) and PMT digitized data. The DB6 design allows nominal running with at least one working downlink and two working uplinks. By means of the FMC connector interface, the KU FPGAs send all the front-end configuration commands and phase-configurable LHC synchronized clocks and receive MB power monitoring information, data from slow current integrator ADCs that sit in the FENICS cards, and samples of two-gains of the signals coming from each PMT digitized by fast ADCs sitting on the MB. Additionally, two buses will allow communication between FPGAs from both halves of the DB, a slow bus (CommBUS) powered by IDDR/ODDR IOs and a high speed 9.6 Gbps bus powered by interconnected GTH transceivers (GTHBUS), allowing signal monitoring, in addition to trigger commands and data interchange between both FPGAs. The DB6 features two Microsemi ProASIC FPGAs to control the remote KU FPGA resets, manage the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

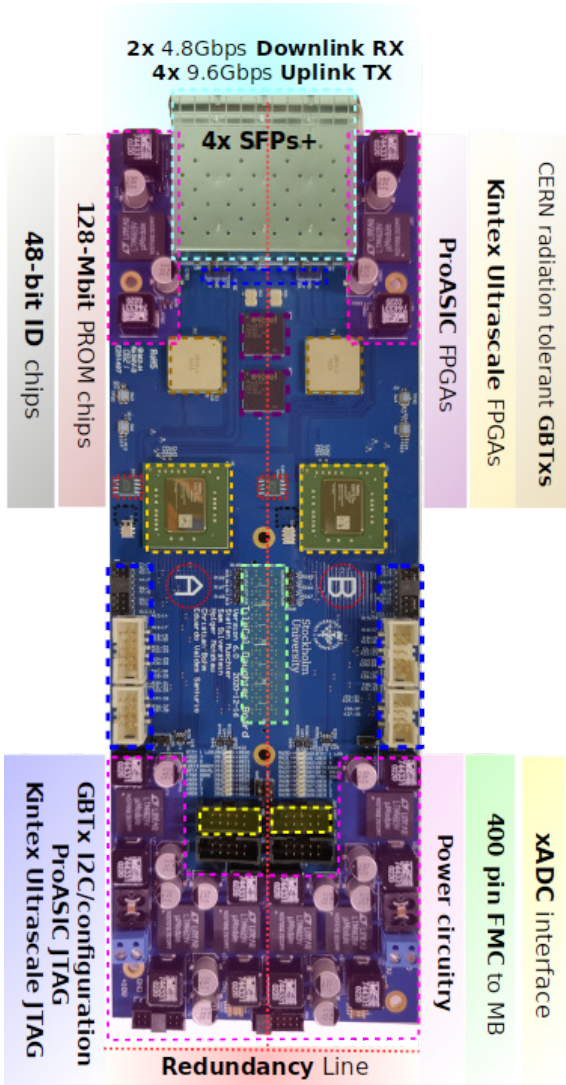


Figure 3: The DaughterBoard revision 6.

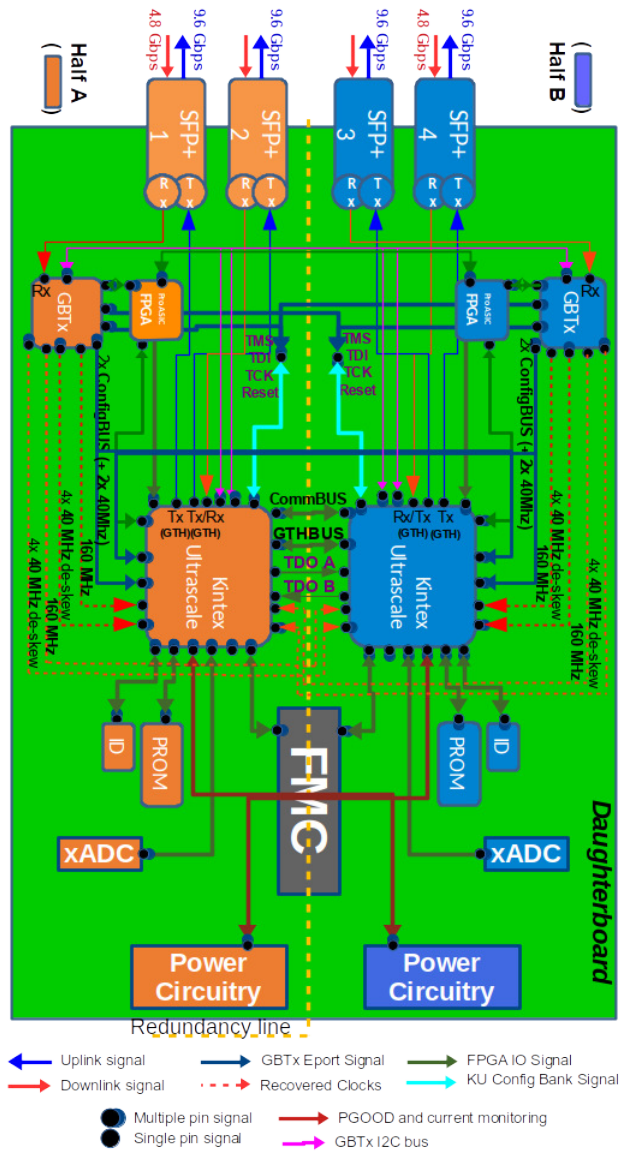


Figure 4: The DaughterBoard revision 6 block diagram.

initialization of the GBTxs and the KU FPGAs, and buffer the JTAG interface signals received from the off-detector by the GBTx that allow remote reconfiguration of the KU FPGAs and their attached FLASH memories. Additionally, the DB6 design includes an xADC header per FPGA for extra analogue sensor monitoring capabilities, and digital ID serial chips for unique MD half identification.

The firmware of the KU FPGAs are in charge of reading out the dual-gain PMT digitized data and formatting it into GBT-CRC protected words. The firmware has a complex clock and timing scheme with multiple clock inputs from various sources such as the two GBTxs, an oscillator and the six ADCs sitting on a MB half. Some of the these clocks are being asynchronously multiplexed depending on the availability of the GBTx links. The DB6 design routing was optimized to take advantage of the Ultrascale dedicated XYPHY BITSLICE byte groups architecture [3]. The bit and flame clocks, and high-gain and low-gain serial data signals coming from each ADC channel will be routed to the same byte XYPHY BITSLICE group. The byte groups used for each specific are 1 or 2, since they contain the dedicated Global Clock Capable IO pins (GCIO) that allow direct access to the Phase Locked Loops (PLLs) and the Mixed-Mode Clock Manager (MMCM) of the bank, and assures that there is no congestion with more than 6 clocks driving IO loads on any half bank. Figure 5 shows the scheme followed for the clock input distribution and the routing of signals of the ADCs and other relevant time-dependent data paths to the different banks. This scheme makes sure that successful timing-closure is easier to achieve during firmware implementation. The DB6 firmware has been under development and extensive testing has taken place in different testbenches and a test beam by interfacing DB6 prototypes to various prototype versions of the TileCal HL-LHC Upgrade system. The experience accumulated from the previous revisions and prototypes of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

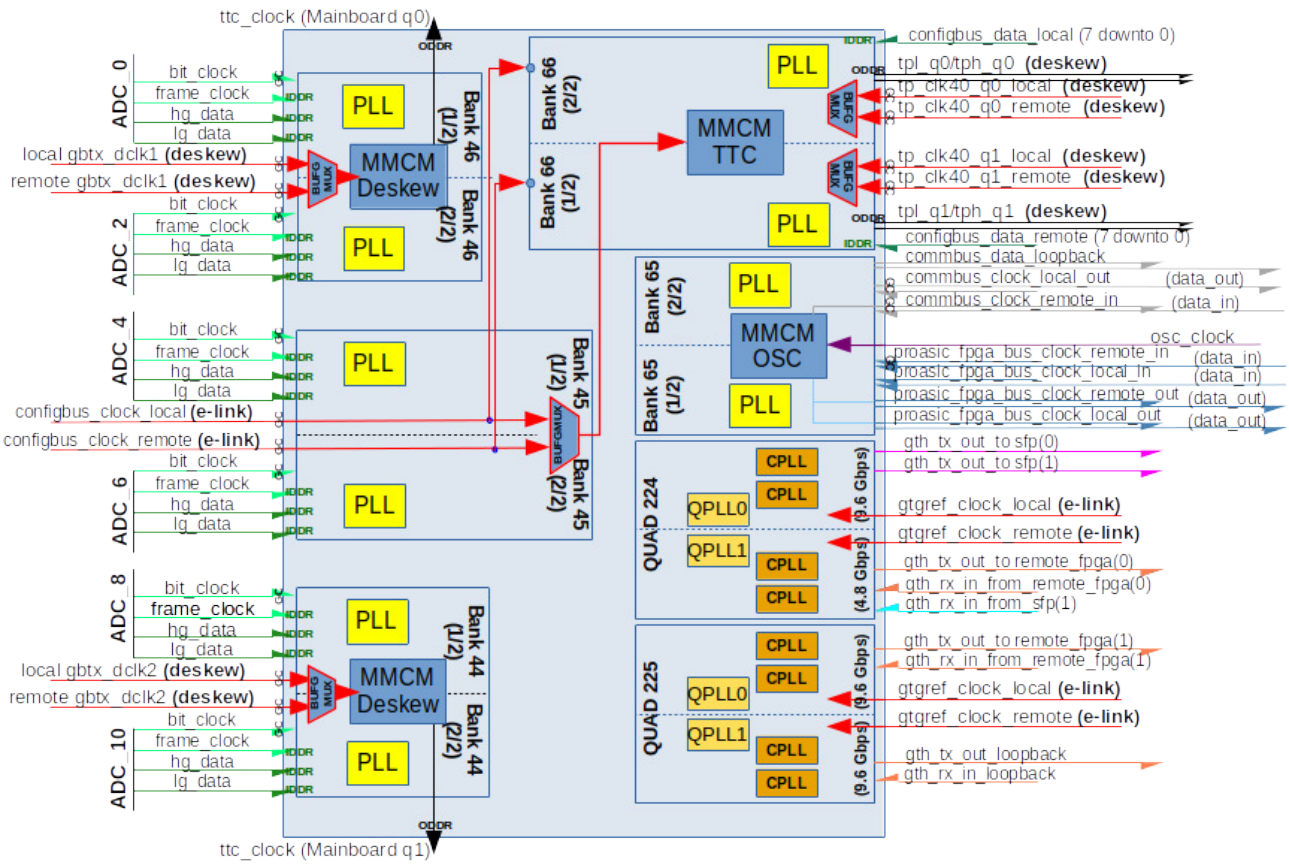


Figure 5: Distribution of clocks, ADC read-out pins (bit_clock, frame_clock, hg_data and lg_data), GTH transceiver signals, Configuration BUS (configbus) interface, TTC phase configurable clocks to drive the digitizing blocks and calibration signals on each MB quadrant (ttc_clock, tpl/tph) and other clock dependent modules for the DB6 Xilinx KU XCKU035 FPGA.

the DB was used as a baseline guide for the DB6 firmware that is currently in active development. Core functionalities have been successfully implemented in the firmware such as stable 4.6 Gbps downlinks and 9.6 Gbps uplinks interface with the TilePPr, ADC read-out, CIS signals, slow ADC Integrator system readout and front-end configurations.

RADIATION TOLERANCE STUDIES FOR THE DAUGHTERBOARD 6 DESIGN

Previous revisions of the DB were tested for TID, NIEL and SEE to verify the radiation tolerance of each iteration for the HL-LHC era. The main driver for selecting the components to us in the board are the Xilinx FPGAs because of their importance in the functionalities of the design and weight in the end-price of the production of the boards. Three generations of the Xilinx Kintex family were studied. The forth revision of the DB (DB4) was powered by 28 nm planar Kintex 7 (K7) FPGAs, the fifth revision by 16 nm FinFET Kintex Ultrascale+ (KU+) FPGAs, and DB6 by 20 nm planar KU FPGAs. The strategy followed for Single Event Upset (SEU) appearances in Xilinx FPGAs rests in using the Xilinx Soft Error Mitigation (SEM) IP Core in the firmware implementation. The Xilinx SEM detects, classifies and corrects SEUs for the configuration memory. Not all the SEUs can be corrected, those classified by the SEM as uncorrectable are planned to be mitigated by using Triple Mode Redundancy (TMR) and partial reconfiguration whenever possible in the firmware, and where none of the previous solutions is possible an FPGA total re-configuration from the attached FLASH memory will be triggered. The K7 FPGAs were sufficient for the radiation requirements of the TileCal for the HL-LHC with manageable SEU rates where manageable with the fluences simulated at the time when the tests were performed [4]. However, there was a need to migrate to a different transceiver architecture because of the bandwidth gap present in the K7 GTX transceivers. It was expected that migrating to smaller feature size such as the ones of KU and KU+ would reduce the cross section for the Single Event Upsets (SEUs) of the FPGA chosen.

The KU+ FPGAs present in the DB5 feature GTY transceivers that allow 9.6 Gbps communication with full compatibility with the LHC timing [5]. The DB5 design proved to withstand up to 20 kRad of TID, being enough to qualify it for the HL-LHC lifetime within the radiation doses simulated as reported in the ATLAS Tile Calorimeter Phase-II Upgrade Technical Design Report [2]. However,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

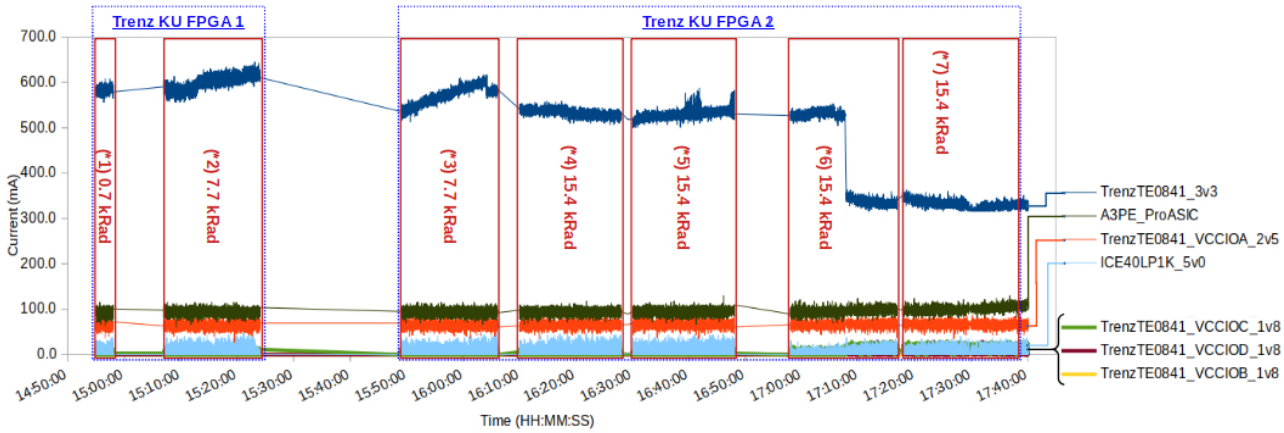


Figure 6: Current monitoring data corresponding to SEL tests with protons at 226 MeV performed on two Trenz TE0841 micromodules (marked as Trenz KU FPGA 1 and Trenz KU FPGA 2), a Microsemi A3PE starter kit, and a Lattice ICEBLINK LP1K evaluation board. The irradiation took place over seven runs, each signalized in the red boxes with (*x) followed the corresponding deposited TID dose. Note that values for the measured currents for TrenzTE0841_VCCIOB, TrenzTE0841_VCCIOC, and TrenzTE0841_VCCIOD are quite close to the current values for the ICE40LP1K_5V0. Therefore, the aforementioned channels are not visible in the plot [6]

.

new dose estimates were obtained by new simulation runs performed with updated geometry. The new values established that to qualify the DB5 design, new TID tests had to performed up to either 72 kRad or to 14.4 kRad if the test included annealing. The SEU tests resulted in 4934 SEUs over a fluence of 2.05×10^{11} protons cm^{-2}, of which only 11 SEUs could not be corrected by the Xilinx SEM. The uncorrectable SEU rates predict that 1.4 $\pm$0.4 uncorrectable errors are expected per DB per year. Between the Xilinx SEM and the TMR it is expected that correctable SEUs will not affect nominal runs. The DB5 SEE tests showed that the KU+ 16 nm FinFET technology is susceptible to Single Event Latch-ups (SEL) with SEL-fluence rate increased from 2×10^{-11} SEL-fluence rate at 58 MeV to 2.36×10^{-10} rate at 226 MeV protons [6]. The over-currents due to the latch-ups did not cause any noticeable damage, however SELs are highly undesirable. Hence, the solutions to mitigate SELs are to add an over-current protection to the design and migration to an SEL-resistant FPGA. Xilinx FPGAs require a specific power-on sequence and the implementation in DB5 was done by means of an RC circuit.

The KU FPGAs present in the DB6 design are powered by GTH transceivers, which are compatible with the 9.6 Gbps communication required by the TileCal HL-LHC Upgrade system. According to the published studies during the time, no SELs were seen in the 20 nm planar Xilinx FPGAs. Two Trenz TE0841 boards (each with the same KU FPGA used in the DB6 design), a Lattice ICE40LP FPGA and a ProASIC3E A3P31500 FPGA were irradiated with 226 MeV protons to verify for SEL sensitivity up to a target fluence of 1.11 10^{12} p/cm^2, covering eight times the fluence simulated for 10-years of HL-LHC. During the test, all the currents of the FPGAs were monitored (Fig. 6) and no SELs were detected. The LCMXO2 CPLDs present in the TE0841 boards for buffering the JTAG chain used to configure the KU FPGAs failed due to TID effects. As a consequence only rough estimates of the SEUs could be obtained through the Xilinx SEM over two runs: 2 uncorrectable errors and 2 333 correctable errors for a rate of 166 soft errors per $10^9 \times$ p/cm^2.

In the case of the ProASIC3E A3P31500, the firmware started failing at around 57 kRad of TID or 75% of the total fluence with no recovery after power cycling. As reported by Actel, the firmware functionality was fully recovered after a long process of annealing and the VPUMP of the FPGA was permanently damaged making the reconfiguration of the chip permanently unavailable [7]. The Lattice ICE40LP FPGA firmware failed after approximately 10^9p/cm^2 of the delivered fluence. The firmware functionality was recovered on-site by reconfiguring the FPGA from the attached FLASH memory after power-cycle.

Even though no SEL occurrences are expected in KU FPGAS, it was decided to integrate an SEL current limiting circuitry in the chain of DC-DC converters used in the power sequence scheme of DB6. This circuitry will power off the whole DC-DC converter chain of the half of the DB where one of the voltages gets a high current drain. This solution will increment the robustness of the board by avoiding potential damages to the FPGAs and other components on the presence of isolated unexpected SELs or overcurrents originated by any power failures. Additional radiation induced problems potentially caused by the use of large electrolytic capacitors at the DC-DC regulators used in previous revisions of the DB are being mitigated by the use of conductive polymer capacitors, which have been observed to be radiation tolerant with no impact to performance after 200 kRad at 500 rad per hour [8].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

The KU FPGA offered a good compromise in SEU rates ranging between K7 and KU+ and no SEL sensitivity up to the HL-LHC simulated fluences. Therefore, two DB6 prototyped were produced to do TID tests to the full design. The two DB6s (DB6-1 and DB6-2) were irradiated by means of a ^{60}Co source at the CERN CC60 facilities. Each of the DBs was exposed to different dose rates and different dose targets. DB6-1 was irradiated to 220 Gy at a "fast" rate of 3.37 Gy/h and DB6-2 was irradiated to 52.6 Gy at a "slow" rate of 0.33 Gy/h. The KU FPGAs and all the board currents were monitored for each of the voltages, putting special interest in detecting any failures on the eight Coretek SFP+, the four KU FPGAs, the four Microsemi ProaSIC3 FPGAs and the reconfiguration FLASH memories. A baseline firmware was programmed in the ProASIC FPGAs and in the KU FPGA attached FLASH memories. After the irradiation period, both DBs were extensively tested in testbenches and none of the aforementioned components were damaged by the deposited TID.

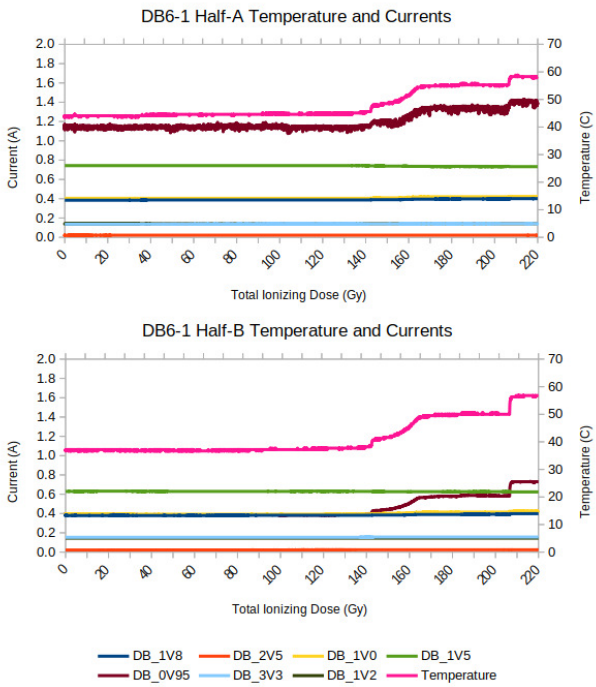


Figure 7: DB6-1 Current monitoring during 220 Gy (22 kRad) of TID exposure at 3.37 Gy/h (0.337 kRad/h).

During the DB6-1 TID exposure (Fig. 7), it was discovered a strong correlations between temperature and current in 0.95 V. An increase in current was measured in the 0.95 V of both sides at around 140 Gy. During the setup a fan was situated to keep the temperatures of the FPGAs steady during the irradiation period. The fan active components started to fail at around 140 Gy, leading to the increase of temperature on the FPGA during the rest of the irradiation period. The 0.95 V supplies the VCCRAM and the VCCINT for the core of the KU FPGA, making the currents drawn

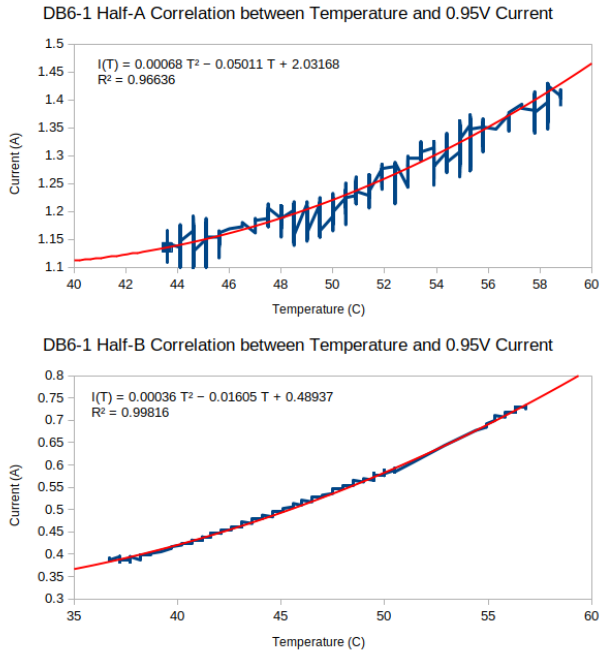


Figure 8: DB6-1 correlation between current and temperature for the 0.95 V channel of both DB halves fixed to a second polynomial degree regression.

from this voltage more sensitive to any temperature change in the FPGA.

A correlation between the 0.95 V current and the FPGA temperature was found by fitting the data to a second degree polynomial regression (Fig. 8). A convolution to the original data was done then to find the expected 0.95 V current behaviour for an exposure with constant temperature (Fig. 9). In each half the temperature value chosen for the convolution was the temperature measured on each FPGA when the test started.

In the case of DB6-2 (Fig. 10), all the currents and the temperatures were stable during the full run.

CONCLUSIONS

The DaughterBoard is the read-out link and control board that interfaces a TileCal MD and the off-detector electronics. The current revision of the DaughterBoard (DB6) fulfills the radiation requirements for TID and SEE imposed by the HL-LHC. The DB6 improved the radiation tolerance of the previous designs that includes the SEL sensitivity issues of DB5, by migrating the design to the non-SEL sensitive 20 nm planar KU FPGA, adding extra protection to mitigate any overcurrent appearances and using conductive polymer capacitors. The studies performed on SEUs showed that even though the KU+ FPGAs offer a better SEU rates, the preliminary rates measured in the KU FPGAs are a good compromise for a design where SELs are unacceptable. To fully qualify the DB6 for the HL-LHC radiation requirements a dedicated SEU test and NIEL tests need to be conducted. Around 930 of the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

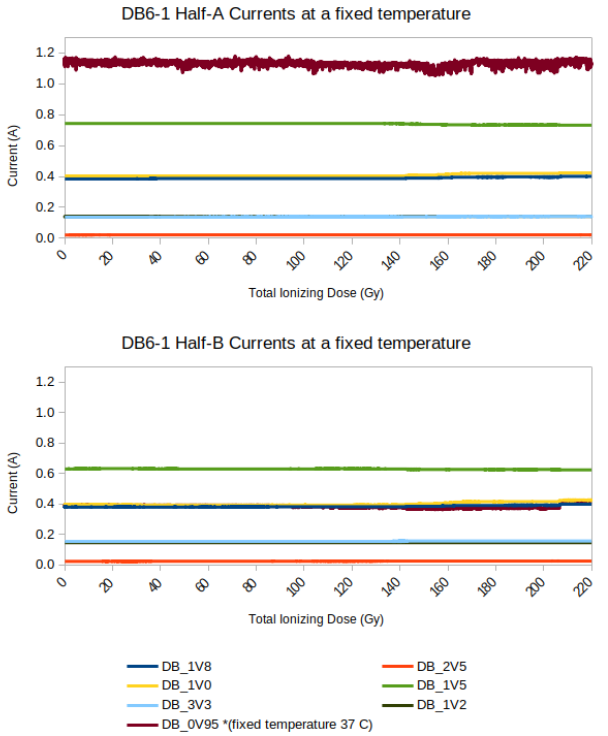


Figure 9: Expected DB6-1 Current monitoring during 220 Gy (22 kRad) of TID exposure at 3.37 Gy/h (0.337 kRad/h) with constant temperature. Only the current measurements for the 0.95 V voltages were corrected for temperature, the correction was done following the initial temperature value measured at the start of the test.

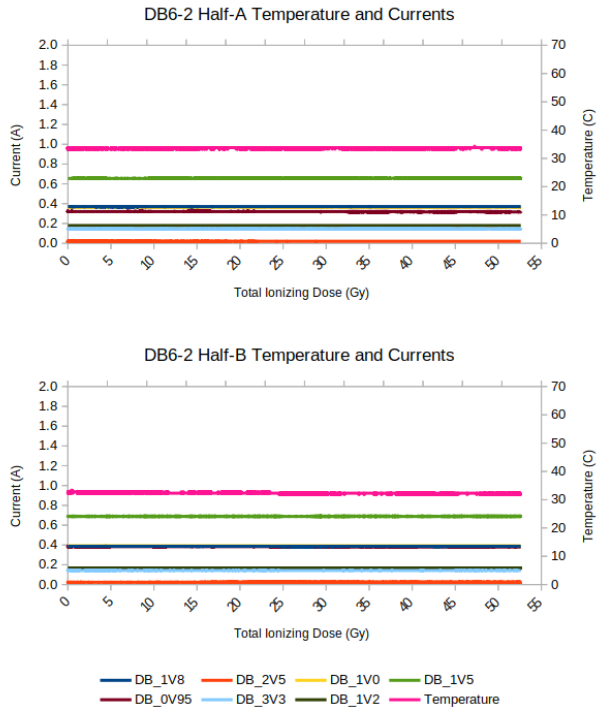


Figure 10: DB6-2 Current monitoring during 52.6 Gy (5.26 kRad) of TID exposure at 0.33 Gy/h (0.033 kRad/h).

redesigned DaughterBoards will be produced for Phase-II as the contribution of Stockholm University to the the ATLAS Upgrade for the HL-LHC era.

REFERENCES

[1] ATLAS Collaboration *et al.*, *Journal of Instrumentation*, vol. 3, p.S08003, August 2008. doi:/10.1088/1748-0221/3/08/S08003

[2] ATLAS collaboration, "Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter", CERN, Geneva, Switzerland, Rep. CERN-LHCC-2017-019, Rep. ATLAS-TDR-028, 2018. https://cds.cern.ch/record/2285583

[3] UltraScale Architecture Clocking Resources User Guide, https://www.xilinx.com/support/documentation/user_guides/ug572-ultrascale-clocking.pdf

[4] H. Åkerstedt1, C. Bohm, S. Muschter, S. Silverstein and E. Valdes, "A radiation tolerant Data link board for the ATLAS Tile Cal upgrade", *Journal of Instrumentation*, vol. 11, p. C01074, January 2016. doi:10.1088/1748-0221/11/01/c01074

[5] ATLAS Tile Calorimeter Link Daughterboard, in *PoS Volume 343 - Topical Workshop on Electronics for Particle Physics (TWEPP2018)*, Antwerpen, Belgium, 2018. doi:10.22323/1.343.0024

[6] Eduardo Valdes Santurio *et al.*, "A Revised Version of the ATLAS Tile Calorimeter Link Daughterboard for the HL-LHC", *IEEE Transactions on Nuclear Science*, vol. 68, no. 9, pp. 2414–2420, Sept. 2021. doi:10.1109/TNS.2021.3103408

[7] Radiation-Tolerant ProASIC3 FPGAs Radiation Effects, https://www.microsemi.com/document-portal/doc_view/131374-radiation-tolerant-proasic3-fpgas-radiation-effects-report

[8] Capacitors for Spacecraft: Withstanding a Harsh Radiation Environment, https://www.ttiinc.com/content/ttiinc/en/resources/marketeye/categories/passives/me-slovick-20160809.html

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR01

UPGRADE OF THE CMS ECAL DETECTOR CONTROL SYSTEM DURING THE CERN LARGE HADRON COLLIDER LONG SHUTDOWN II*

L. Marchese†, D. Di Calafiori, G. Dissertori, L. Djambazov, R. J. Estupiñán, W. Lustermann,

ETH Zürich, Zürich, Switzerland

Abstract

As part of the Compact Muon Solenoid (CMS) experiment, the Electromagnetic Calorimeter (ECAL) Detector Control System (DCS) is undergoing a large software and hardware upgrade during the second long shutdown (LS2) of the CERN Large Hadron Collider (LHC). The DCS software running under the WinCC Open Architecture (OA) platform, required fundamental changes in the architecture as well as several other upgrades on the hardware side. The extension of the current long shutdown (2019-2021) is offering a unique opportunity to perform more updates, improve the detector safety and robustness during operations and achieve new control features with an increased modularity of the software architecture. Starting from the main activities of the ECAL DCS upgrade plan, we present the updated agenda for the LS2. This covers several aspects such as the different software migrations of the DCS, the consolidation of toolkits as well as some other improvements preceding the major ECAL upgrade foreseen for the next long shutdown (2025-2026).

INTRODUCTION

The CMS (Compact Muon Solenoid) [1] experiment at the CERN Large Hadron Collider (LHC) is a general multi-purpose detector designed primarily to probe proton-proton and heavy ion collisions. The detector is built around a huge superconducting solenoid of 6m internal diameter providing a magnetic field of 3.8 T. A silicon pixel and strip tracker, a lead tungstate crystal electromagnetic calorimeter and a brass scintillator hadron calorimeter are located within the solenoid volume. Muon detectors are embedded in the steel flux-return yoke outside the solenoid.

The electromagnetic calorimeter consists of about 76,000 $PbWO_4$ scintillating crystals and a lead/silicon preshower. The detector is conventionally sub-divided in three main partitions: Barrel (EB), Endcaps (EE) and preshower (ES). The EB is organised in 36 super-modules forming a cylinder around the proton-proton interaction point. Each super-module contains 1700 crystals arranged in four modules. The EEs are the structures which close both ends of this cylinder with each of them formed by two half disks named DEEs and containing 3662 crystals. The ES consists of two circular structures placed in front of the EEs. More details on the CMS ECAL detector can be found in [2].

The challenging constraints on the design of the ECAL required the development of a complex sophisticated Detector Control System (DCS). The ECAL DCS has successfully supported ECAL operations since the commissioning phase of the detector contributing to an efficient data collection during Run 1 and Run 2 operations. Detector maintenance at CMS closely follows the LHC calendar with major upgrades postponed to special times, known as the Extended Year-End Technical Stops (EYETS) and the LHC Long Shutdowns (LS). We are currently in the second major long shutdown LS2 [3] since the start of the LHC. Initially planned to last two years until April 2021, due to the ongoing COVID-19 pandemic the LS2 was extended by one year. During the LS2 both hardware and software upgrades were performed, as described in [4]. In this paper we focus on the software upgrades, including the software migration which allowed the ECAL DCS to be up-to-date with the latest versions of the control platform and frameworks.

This paper is organized as follows. After describing the CMS ECAL DCS, we describe the software upgrade performed during LS2 where different upgrades are reported in four subsections. An additional section reports on the reorganization of the notification system of the ECAL DCS, which also took place during LS2. Finally, we summarize the contents in the last section.

THE CMS ECAL DCS

The ECAL DCS was designed to ensure an autonomous control and monitoring of the working conditions of the ECAL detector and to guarantee the detector is properly powered and able to collect data when the LHC is operational. The ECAL DCS is organised according to the detector services it provides, where the latter can be categorized in four different groups: powering services, safety services, environmental monitoring services (humidity and temperature) and external services. The ECAL DCS architecture is schematically shown in Fig. 1.

The DCS software supervises the interaction among the different subgroups mentioned above and runs on three DELL blade servers installed with the Windows Server 2008 R2 operating system. A redundant software replica is also available in the event of a critical failure of the primary system. The DCS software is assembled via the commercial WinCC Open Architecture (WinCC OA) control system toolkit from ETM [5] along with CERN software frameworks known as JCOP Framework [6] and components developed by the Central CMS DCS team [7]. Industry standards, as OPC Data Access (OPC DA), Modbus and

* on behalf of the CMS Collaboration
† luigi.marchese@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

S7 protocols for hardware communication are used. As part of such ECAL DCS migration described in this paper, the OPC DA servers have been migrated to OPC Unified Architecture (UA) servers.

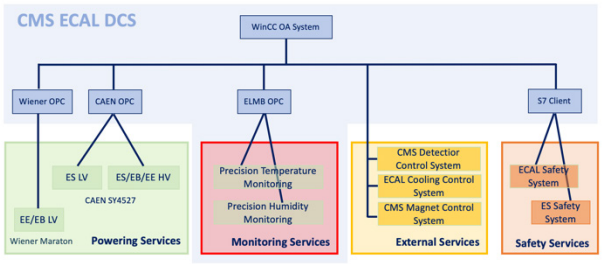


Figure 1: ECAL Detector Control System (DCS) architecture including its connection with other services. Inspired by [5].

Commands are sent to the ECAL subsystems via a Finite State Machine (FSM) [9], the core of each control system at the LHC. Thanks to a logical grouping of DCS subsystems into a hierarchical tree-like structure, the FSM enables a simplified representation of the DCS by introducing a set of defined states for each detector partition, for instance state ON/OFF for the ECAL EEs, and specific rules to move from one state to another. Within this architecture, the state of each sub-component is clearly summarized at higher level and high-level commands can be propagated down to individual ECAL partitions in a controlled way. This allows process variables to be summarized in a single-human-readable state and control commands to be implemented by a shifter in the CMS Control Room without a detailed expertise of the underlying hardware.

SOFTWARE MIGRATION

During the current long shutdown, the ECAL DCS had to go through some software migrations which can be subdivided into three types: WinCC OA and JCOP control framework migration, OPC UA migration and control system migration. They are described in more detail in each of the subsections below. Following these migrations, an additional review of the DCS software was also needed, as reported in the final part of this section.

WinCC OA and JCOP Control Framework Migration

This migration was centrally coordinated by the CMS DCS team, as part of a CMS-wide plan to accommodate all the CMS DCS software with the latest release of the control platforms. The plan was discussed and agreed internally in CMS in the first half of 2020. The final CMS migration took place in July 2021 with a few validation tests performed in April and June 2021 during a few days of ECAL operations.

The ECAL DCS was migrated from WinCC OA version 3.15 to version 3.16 and from the CERN JCOP control framework version 8.1.2 to version 8.4.0 [10]. WinCC OA migrations at CERN are common, as the software needs to be up to date with the latest versions ETM provides support for. Nevertheless, this specific migration required a major effort as the new version of the platform only supports a specific encoding type forcing a migration from the old type.

WinCC OA Migration. While projects in WinCC OA versions lower than 3.16 were created using the ISO-8859-1 (Latin) encoding by default, the 3.16 version only supports Unicode UTF-8 encoded projects. This is part of an attempt by ISO and the Unicode Consortium to develop a coding for electronic text that includes every existing alphabetic symbol, since the ISO-8859-1 (Latin) encoding covers ASCII characters but doesn't include several symbols frequently used in other languages different from American English or some of the most popular western European languages. Compared to the ISO-8859-1 encoding, where each symbol is represented by a single byte (character data type), in the UTF-8 project each character is represented by a multibyte encoding with 1-4 bytes. Since only a sub-set of the Unicode UTF-8 encoding is backwards compatible with the previous encoding, the ECAL DCS project required a review of all the codes to check if they were compatible with such a change. ECAL DCS source files were checked looking for non-compatible characters. Automatic tools were implemented to detect and update pieces of codes. As a result, only 11 of the 304 ECAL DCS files required such changes. They were often code comments written by international developers using special characters from languages different from American English.

While analysing the ECAL DCS codes, a special care was devoted to some strings which needed to be treated as arrays of characters for some applications. Since individual characters of these arrays must be processed as such for a proper functioning, a search for these specific arrays of characters through the ECAL DCS codes was required. A program has been developed searching for these special character-type situations, like character-type declarations or arrays of characters, allowing the discovery of pieces of codes which would have failed otherwise.

An additional effort was performed to develop a new set of Unicode standard functions, which were also ASCII-compatible. In total, the code was updated to accommodate the introduction of 11 new types of functions in the ECAL DCS project. In order to reduce transitioning periods between the two WinCC OA versions, all the ECAL DCS components were upgraded already in the legacy system 3.15 so that the development proceeded in parallel for both versions.

With respect to user's interface, special characters in the panels might not be displayed correctly if the panels were written with ASCII-compatible characters. In large projects it happens often that some elements from old panels are copied and pasted into new panels. When importing pieces of code, such as buttons etc., from panels written with the ISO encoding into new panels written with the UTF encoding, some artefacts might be generated. ETM recommended that in order to convert existing panels to the new UTF format, a conversion to *XML* files should be performed. A new XML file format for storing purposes of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR01

project panels has been adopted in WinCC OA version 3.16. Even though the previous file format could have been compatible, for consistency it was decided to convert all the existing files in the ECAL DCS system accordingly. Being the XML format also compatible with the previous WinCC OA version, an earlier deployment was already implemented in the previous WinCC OA 3.15 legacy system. As the conversion was completed, the converted panels were re-analysed via a set of tools to remove redundant declarations and correct for possible undetected conversions.

JCOP Control Framework Migration. The Joint Control Project JCOP is a collaboration between the CERN LHC experiments, the EP Department and the Industrial Group in the Beams Department at CERN aiming for a shared control system architecture used LHC-wide. As part of this mission, in 2003 the group released a common JCOP framework which allowed the configuration, the monitoring and the operation of different sub-detectors of the LHC experiments enabling communication with the Data Acquisition systems and the CERN infrastructure services, as well [11]. Updated versions are produced periodically. Following the new release of the WinCC OA platform, the JCOP group released a new version of the JCOP framework compatible with it. Thus, the ECAL DCS had to migrate the framework used in the legacy system version 8.1.2 to 8.4.0. In the previous JCOP framework versions most functions worked with no need for an explicit call of required libraries, as libraries and utilities were part of a global scope of WinCC OA. Due to performance purposes, this software design was abandoned starting from version 8.4.0 of the JCOP framework causing a review of all the source codes to explicitly call required framework libraries. This review affected not only the ECAL DCS system, but almost all the DCS systems implemented by the other CMS sub-detector groups. Given the large use of the past functionality, this required a major effort. Thus, the JCOP and CMS Central DCS teams provided the sub-detector DCS teams with specific tools for an automatic identification and modification of sources which required an update. As recommended by the JCOP framework team, the tool codePurger has been used to scan the entire ECAL DCS project searching for unresolved calls of functions and implementing automatic modifications. This resulted in the creation of a reference database which was later used for such identification and modification of sources requiring an explicit call for libraries. In total 240 code transformations were required, out of 120 source files analysed by the tool. For consistency the updated codes were also compared to previous versions proving the reliability of the tool and implemented updates.

OPC Unified Architecture Migration

Since 1996 the OPC DA standard has been successfully allowing communications between any hardware regardless of its vendor. At CERN, OPC DA has been heavily used to control and monitor a large variety of devices, which are controlled by a SCADA layer provided by WinCC OA. WinCC OA communicates via its built-in OPC driver - an OPC Client - to an OPC server, which communicates with the physical device. A schematic representation is shown in Fig. 2. Following the release by the OPC foundation of the OPC-UA specifications and implementations, CERN decided to migrate all the existing OPC DA servers to OPC UA [12]. Compared to the old OPC DA based on the deprecated Microsoft COM/DCOM, OPC UA is now based on modern industry standards making it a more compelling candidate for controlling and monitoring middleware between SCADA systems and hardware devices.

Some of the most important OPC UA highlights include:

- its platform independence being not tied anymore to Microsoft platforms,
- its improved security since it has a comprehensive security model built in securing channel client/server communication,
- its improved modelling since it has an extensive vocabulary for modelling purposes which allows to type components and express inter-component relationships.

Within the WinCC OA – OPC architecture described above, hardware addresses are encoded in the form of datapoints. Individual software components for each application are responsible for the final installation and configuration of datapoints related to that application.

The OPC UA migration in CMS took place in parallel with the WinCC OA/JCOP framework migration during summer 2021 with several validation tests performed in spring 2021. The ECAL DCS makes use of three different OPC servers to monitor and control the following devices:

1. Wiener Marathon power supplies for EE/EB Low Voltage (LV) via Wiener OPC,
2. CAEN power supplies for EE/EB High Voltage (HV) and ES LV and HV via CAEN OPC,
3. CERN-made module Embedded Local Monitor Boards (ELMBs) used for ES High Voltage monitoring, for EEs HV and for ECAL Precision Temperature Monitoring via CANOpen OPC.

The software components affected by this migration in ECAL DCS are reported in Fig. 2. Every component required a reconfiguration of the address space. This was done via an automatic tool released by the JCOP framework team for the different OPC servers. This migration also allowed for a conceptual change from a monolithic single-server configuration to a multiple-server paradigm resulting in a further splitting of the current OPC server's configuration. Benefiting from this, it was possible to anticipate some of the granularity changes which will be required for the LS3 upgrade, when the ES and EE detectors will be removed and so the related OPC UA servers. Having a more granular architecture already in place will simplify the removal operation expected for LS3.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

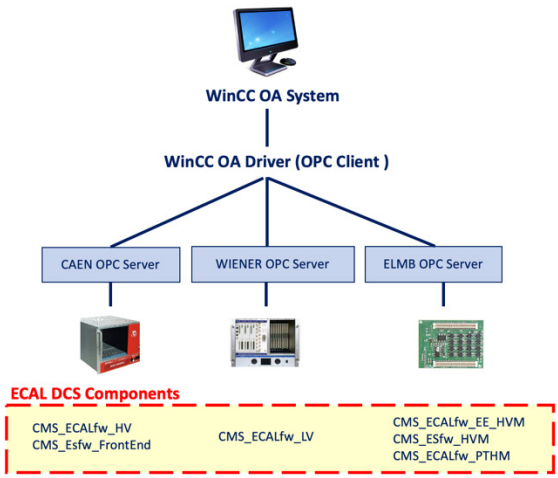


Figure 2: WinCC OA/OPC architecture used at CERN with the indication of the three different servers in use. For each type the ECAL DCS component, which had to undergo the OPC UA migration, is also indicated.

Version-control-system Migration

After having been largely used during LHC Run 2 operations, the Apache Subversion system (SVN) [13] for versioning and revision control system was officially abandoned by CERN in 2018. This forced all the LHC experiments to migrate their SVN-based projects to the Git distributed version control system, hosted by the CERN GitLab service [14]. The ECAL DCS team took this opportunity to also implement some hierarchy reorganization of the project repositories. When moving to Git repositories, the original SVN repository was split into a hierarchy of individual repositories, which were structured according to their functionality. As a result, the ECAL DCS system moved from a single repository to a set of 35 different repositories.

Additional Reviews

In this section we describe a few additional software reviews, which were needed following the migrations reported in the previous sub-sections.

When installing a specific component, some other components might be needed prior to the installation of the component in consideration. This is due to the dependency chain which must be respected for the component to be installed properly. The JCOP migration described in the previous section affected the dependency chain to be followed when installing a component. As for the other CMS DCS projects, the installation of the ECAL DCS components follows guidelines from the JCOP framework group. When installing a component, the component itself should provide all the required information respecting the needed dependencies. A specific toolkit, the ECAL DCS installation toolkit, has been developed over the past years for such a purpose. Following the software migration and the introduction of new libraries, the toolkit has been reviewed and modified to meet some new requirements, such as the new configuration of OPC UA servers.

An additional software review was also needed regarding the new OPC UA configuration for Wiener OPC. Following a hardware replacement for ECAL LV (which took place during data-taking operations in August 2021), a type mismatch in the address configuration between OPC UA and WinCC OA has been discovered. After running some tests, the default type in WinCC OA was moved from bool32 to int32 solving the mismatch. This was not observed during the validation tests in spring 2021 and became clear only after the whole migration took place. It was also reported to the CERN OPC experts for future improvements.

UPDATED NOTIFICATION SYSTEM

In parallel to the software migration, during the LS2 some work was also done to improve the current notification system. In the ECAL DCS project, components use a central tool developed by the Central DCS team for configuring alarm-based notifications. By using this tool, the system accesses the needed CERN e-group – the interface developed at CERN to manage groups – and import a list to distribute messages to its members. The ECAL DCS team has contributed to some improvements for a further e-group integration. While in the past the tool required periodic synchronizations to be fully operational, this is not needed anymore for the new version to operate. Specifically, in the past ECAL DCS notifications and recipients were part of groups – the notification groups – whose lists of recipients were also synchronized once or twice a day. Since the notification groups were used by different components to send messages, the previous centralized system allowed for multiple messages to be sent to the same group of users. During ECAL operations this might be problematic, as it becomes hard to monitor new alarms among several replicas of others. A reorganization of the notification system was designed in 2020. The new system is organised by application domains, with each of them used exclusively by a single component. This prevents the possibility of multiple messages to be sent by different components. Users can then subscribe/unsubscribe directly to an independent e-group for a specific notification domain simplifying the overall system.

CONCLUSION

After several successful years during Run 2 operations at CMS, the CMS ECAL DCS project is undergoing major modifications during the current second-long shutdown at the LHC. During summer 2021, system codes have been reviewed and modified to accommodate several software migrations ongoing in CMS, such as the WinCC OA and JCOP control framework migrations. In parallel, OPC DA servers used by the ECAL DCS project have been migrated to the new OPC UA configuration. Taking advantage of the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR01

long shutdown, additional upgrades were also performed. All the code repositories stored since the start of the ECAL DCS project have been move to the CERN Gitlab version control system and the ECAL DCS notification system has been reorganised for a more efficient functionality.
Some validation tests were performed in spring 2021, in preparation for the final migration which took place during summer 2021. The ECAL DCS system has been successfully migrated and will serve as the ECAL DCS legacy system for the Run 3 data-taking operations starting in March 2022.

ACKNOWLEDGEMENTS

The ECAL DCS project is supported by the Swiss National Science Foundation, Switzerland. The authors would like to thank also the CMS Central DCS group and the CERN JCOP framework group for their support through several upgrades described in this paper.

REFERENCES

[1] CMS Collaboration, "The Compact Muon Solenoid, Technical Proposal", CERN-LHCC-94-038 LHCC-P1, CERN, Switzerland, December 1994.

[2] CMS Collaboration, "The CMS electromagnetic calorimeter project: Technical Design Report", CERN-LHCC-97-033 CMS-TDR-4, CERN, Switzerland, 15 December 1997.

[3] M. Bernardini, K. Foraz, "Long Shutdown 2 @ LHC", in *Proceedings of the Chamonix 2014 Workshop on LHC Performance*, Vol. 2, CERN, Switzerland, 2015.

[4] R. J. Estupiñán et al., "CMS ECAL Detector Control System upgrade plan for the CERN Large Hadron Collider Long Shutdown II", in *Proceeding of the 12th Int. Workshop on Emerging Technologies and Scientific Facilities Controls (PCaPAC'18)*, Hsinchu, Taiwan, Oct. 2018.

[5] ETM ,http://www.winccoa.com/company

[6] O. Holme et al., "The JCOP Framework", in *Proceedings of the 10th International Conference on Accelerator and Large Experimental Physics Control Systems*, WE2.1-60, CERN, Switzerland, 10-14 October 2005.

[7] R. Gomez-Reino et al., "The Compact Muon Solenoid Detector Control System", in *Proceedings of the 12th International Conference on Accelerator and Large Experimental Physics Control Systems, MOB005*, Kobe, Japan, 12-16 October 2009.

[8] O. Holme et al., "Preparing the hardware of the CMS Electromagnetic Calorimeter control system and safety systems for LHC Run 2", in *Proceeding of the Topical Workshop on Electronics for Particle Physics 2015*, Lisbon, Portugal, Oct. 2015.

[9] C. Gaspar, B. Franek, "Tools for the Automation of Large Distributed Control Systems", in *Proceedings of the 14th IEEE-NPSS Real Time Conference*, Stockholm, Sweden 4-10 June 2005.

[10] R. J. Estupiñán et al., "Software migration of CMS ECAL Detector Control System during the CERN Large Hadron Collider Long Shutdown II", in *Proceeding of CHEP2021*, EPJ Web of Conferences 251, 04007, 2021.

[11] CERN JCOP Framework, http://jcop.web.cern.ch/jcop-framework

[12] B. Farnham, R. Barillère, "Migration from OPC-DA to OPC-UA", in *Proceedings of ICALEPS2011, MOPMS025*, Grenoble, France 2011.

[13] Apache Software Foundation Subversion, http://subversion.apache.org

[14] CERN Gitlab, http://gitlab.cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR03

THE CONTROL SYSTEM OF THE LINAC-200 ELECTRON ACCELERATOR AT JINR

A. Trifonov*, M. Gostkin, V. Kobets, M. Nozdrin, A. Zhemchugov, P. Zhuravlyov,
Joint Institute for Nuclear Research, Dubna, Russia

Abstract

The linear accelerator Linac-200 at JINR is constructed to provide electron test beams with energy up to 200 MeV to carry out particle detector R&D, to perform studies of advanced methods of beam diagnostics, and to work as an irradiation facility for applied research. While the accelerator largely reuses refurbished parts of the MEA accelerator from NIKHEF, the accelerator control system is completely redesigned. A new distributed control system has been developed using the Tango toolkit. The key subsystems of the accelerator (including focusing and steering magnets control, vacuum control system, synchronization system, electron gun control system, precise temperature regulation system) were redesigned or deeply modernized. This report presents the design and the current status of the control system of the Linac-200 machine.

INTRODUCTION

Linear electron accelerator Linac-200 (Dzhelepov Laboratory of Nuclear Problems, JINR, Dubna, Russia) is a unique facility intended for scientific and methodological research in the field of accelerator physics and technology, elementary particles detectors research and development, as well as fundamental and applied research in the fields of materials science and radiobiology. It is based on the MEA linear electron accelerator which was transferred to JINR from NIKHEF in the end of 90s.

Main accelerator structure unit is a station. The injector station A00 includes the electron gun, chopper, prebuncher and buncher. First accelerator station A01 includes one accelerating section and a klystron, which also feeds the RF equipment of the A00 station. All the rest stations include two accelerating sections and a klystron each.

Current setup (Fig. 1) consists of 5 stations, A00–A04, and allows generation of the 200 MeV electron beam. In the future it is planned to increase the number of stations up to thirteen, and the energy will accordingly increase up to 800 MeV [1].

Almost all MEA equipment is in good condition and has reasonable operating resource. However, control systems hardware and software, as it is the most rapidly developing sphere, were mostly out-of-date already at the moment of accelerator transfer from NIKHEF to JINR. Therefore, two major accelerator control system upgrades took place. The first one was continuous, when necessary control subsystems were developed when they were needed [2]. The second one started in 2018 when development of the new global control system began [3].

The global control system automatically collects, processes, stores and displays information about the operation of the accelerator, as well as about the state of the technological equipment involved in the operation of the accelerator. Also, the global control system provides the ability to automatically control individual accelerator subsystems.

The main requirements for the Linac-200 global control system is high reliability, safety, simplicity of software development, and ease of technical support. There should be an opportunity for future modifications and extensions. The software of the global control system should use standard interfaces for interaction between components and be able to use existing developments of the world community.

Tango Controls allows to create control systems that meet these requirements. It is also worth mentioning that the control system for the NICA collider at JINR is being developed using Tango [4]. Therefore, it was decided to implement a new global control system for the Linac-200 accelerator based on Tango.

CONTROL SYSTEM CONCEPT

Tango Controls

Tango Controls is a free open source device-oriented controls toolkit for controlling any kind of hardware or software and building supervisory control and data acquisition (SCADA) systems. The fundamental unit of Tango is a device, which is an abstraction hiding real equipment or program component behind the standard interface. Tango provides high level client application interface which has necessary programming classes to implement client-server communications – synchronously or asynchronously execute commands, read or write attributes, or use events to acquire the data from the Tango devices. Tango incorporates a number of tools to build efficient control system environment including centralized administration and monitoring, access control, logging system, data archiving and code generation for rapid development of the Tango device servers using C++, Java and Python [5].

* trifonov@jinr.ru

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR03

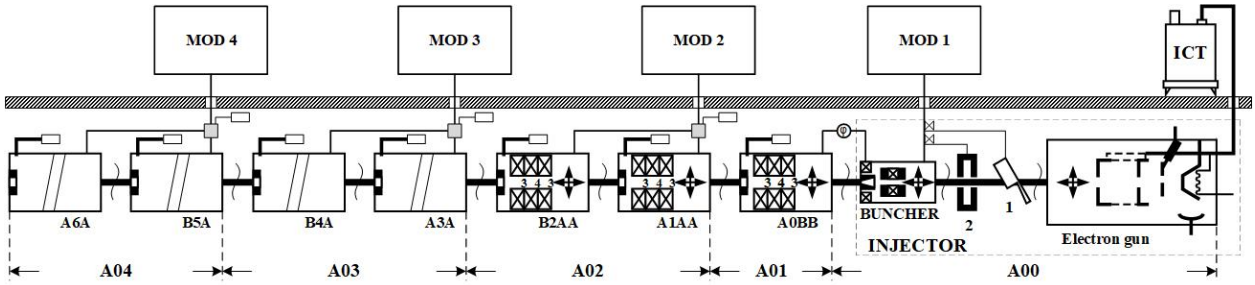


Figure 1: Linac-200 accelerator layout (1 – chopper, 2 – prebuncher).

Control System Structure

Layout of the Linac-200 control system is shown in Fig. 2. The system is implemented as three levels: hardware access level, server level and client level.

At the hardware access level, digital and analog signals are collected and analog signals are converted to digital. This level also includes all equipment that needs to be controlled. The hardware access layer is connected to the server layer through a data transmission channels. Ethernet and RS-485 are used as data transmission interfaces.

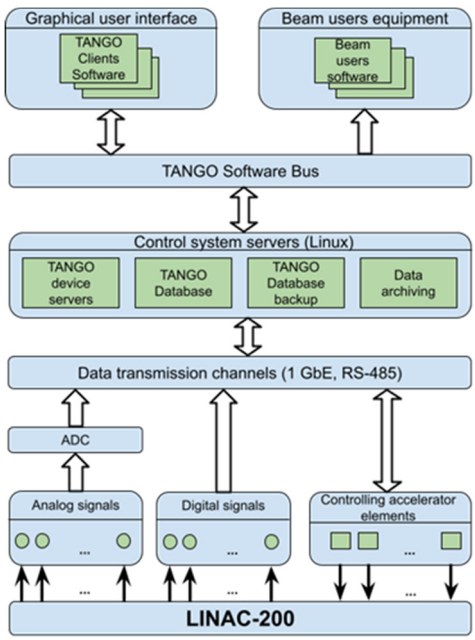


Figure 2: Linac-200 control system layout.

The server level is required for the effective centralized collection, processing and storage of data from the elements of the hardware access level, as well as for the implementation of subsystems control algorithms.

The server level provides a set of software components that are necessary for the effective functioning of the control system. These components include:

- Tango database (for which it is necessary to provide reliable backup methods, the ability to quickly transfer to another server).
- System of differentiation of access rights to the elements of the control system.
- A data archiving system that allows you to store data in a database for short-term or long-term storage.
- A data archiving system that allows you to store data in a database for short-term or long-term storage.

Supermicro [6] servers running Linux Debian are used as server computers. C++ and Python languages are used for server software development.

The client layer provides graphical user interface (GUI) for monitoring and controlling various parameters of the accelerator. At the moment, in the Linac-200 control system, client applications are developed in C ++ using Qt5 and QTango [7]. For each subsystem (or, if there is such a need, for individual elements of the subsystem) there is a separate client application. In addition, there is a global client application that provides a graphical interface for launching client applications for individual subsystems (or their elements).

In the future, it is possible to use web clients, which can be implemented using the technology described in [8]. Also, the possibility of using Taurus [9] – a framework for creating both GUI and command-line applications with Python, seems promising.

Network Infrastructure

The Linac-200 global control system is based on the interaction of components included in the Ethernet network, divided into a number of segments serving the main control room, accelerating stations, users control room and accelerator hall. The network infrastructure of the global control system is shown in Fig. 3.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

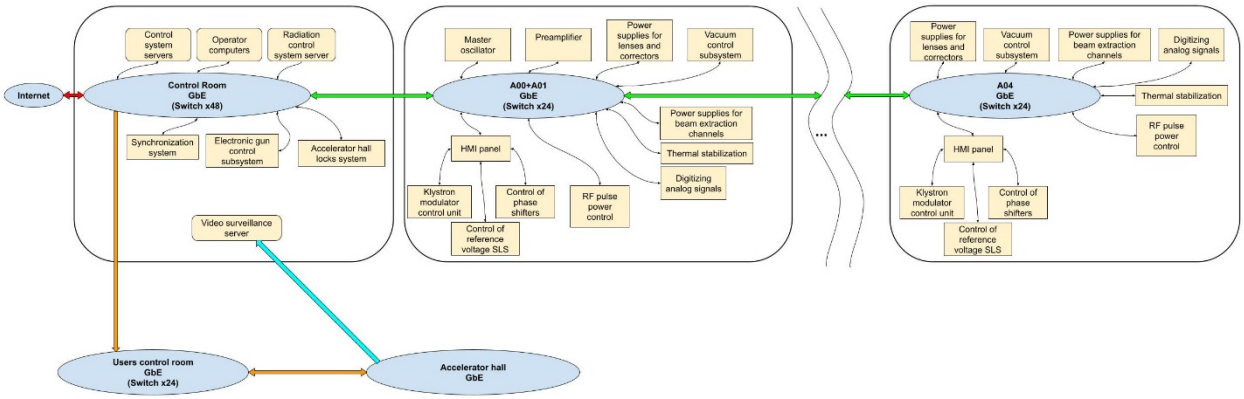


Figure 3: The network infrastructure of the Linac-200 global control system.

Local Control

Local control is available for a number of subsystems and is done with the Weintek MT8071iP [10] operator panel.

In addition to displaying the local control graphical interface, the Weintek MT8071iP panel acts as a Modbus server. Slave devices are connected to the panel by means of RS-485 interface according to Modbus RTU protocol. Data is transmitted to the Tango-based global control system via Ethernet using Modbus TCP and Modbus RTU over TCP protocols.

LINAC-200 CONTROL SYSTEM SUBSYSTEMS

Electron Gun Control System

The beam is generated by the 400-kV DC triode-type electron gun with a thermionic cathode. As part of the development of a global Tango-based control system, new software for the electron gun control was developed.

More information about electron gun control system can be found in [11].

Synchronization System Control

The new synchronization system is a multichannel generator that generates sync pulses to control the accelerating system and synchronize the accelerator subsystems. It consists of a master oscillator and a multichannel delayed pulse generator.

At the moment, a separate software running on a Windows computer is used to control the synchronization system. In the future, it is planned to integrate the synchronization system software into the global Tango-based control system. For this purpose, a corresponding Tango class will be implemented, as well as a client application with a graphical interface.

Control of RF system elements

At the moment the following RF system elements are controlled:

- Master oscillator.
- Preamplifier.
- Klystrons modulator control units.
- Phase shifters.

Master oscillator. A high-frequency signal generator AKIP-7SG384 is used as a master oscillator. The interface for interaction with the control system is Ethernet. Data exchange via Ethernet is carried out using the Tango Socket device [12]. The control of the frequency and amplitude of the output signal is realized.

Preamplifier. A specially designed power amplifier DIALTEK UMP245-300 is used as a preamplifier of the RF system. The control of the output power and RF pulse width is realized. Data exchange between preamplifier and the control system is implemented using Modbus RTU over TCP protocol.

RTU over TCP provides a method to transmit RTU messages with a TCP/IP wrapper through a gateway onto Ethernet without changing any of the bytes in the message. Standard Modbus RTU is meant for transmission over serial lines. The message starts with a one byte Slave ID and ends with a two byte CRC (Cyclical Redundancy Checking). In RTU over TCP, the server does not have a slave ID and uses an IP address instead.

Since the existing Tango class [13] does not support this protocol, a new TangoPyModbus class to handle the Modbus protocol was implemented base on the PyModbus library [14].

Klystrons modulator control units. The solid state modulator consists of the special multicore pulse adding transformer and 40 PFN (Pulse Forming Network) units. Each of these units is a printed circuit board with a 2 kV line-type (50 μs) modulator. Two units feed a primary of the pulse adding transformer [15]. Such a layout allows generation of pulses which amplitude can be changed by changing the number of active PFN units. This number is controlled by the special Weltek controllers. These controllers also provide information on the modulator status to the control system [16]. Interaction with global control system, as well as the local control is realized by means of Weintek MT8071iP operator panel.

Phase shifters. A specially designed mechanical phase shifter (Fig. 4) is used to control the phase shift of the klystron actuating signal of the Linac-200 accelerator station. It has a high resolution positioning of the sliding coaxial line along the length, due to the use of a stepper motor and high manufacturing precision of the ballscrew.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR03

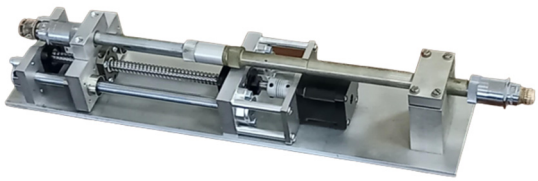


Figure 4: Mechanical phase shifter.

The phase shifter is controlled by means of a specially designed electronic board, the main elements of which are a microcontroller and a stepper motor drive.

Interaction with global control system, as well as the local control is realized by means of Weintek MT8071iP operator panel.

Focusing and Steering Magnets Control

Magnetic elements are used to focus and steer the position of the beam, which are mainly required to reduce the loss of particles during acceleration. Focusing is carried out using solenoidal and quadrupole lenses. Steering magnets are used to adjust the position of the beam relative to the accelerator axis.

The magnetic elements are powered by the KORAD KA3005P and KA6003P power supplies. Modifications (KA3005P / 6003P) differ in the range of currents and voltages (0–30 V, 0–5 A and 0–60 V, 0–3 A, respectively) and are selected based on the power consumed by a particular element of the accelerator magnetic system. Remote control interfaces include USB and RS-232.

The integration of these power supplies into the global Tango-based control system is hampered by the fact that the USB and RS-232 interfaces are unsuitable for transmitting data to the control room of the accelerator (a distance of about fifty meters), and the built-in software of the power supplies does not allow each unit to be assigned a unique address for its unambiguous identification.

To solve this problem, intelligent interface modules KPI-11 were developed. Based on these modules, a control system for magnetic elements was developed (the diagram is shown in Fig. 5). This system allows to remotely and independently control the network of power supplies KORAD KA3005P / 6003P [17].

Vacuum Control System

The diagram of the vacuum control system is shown in Fig. 6.

The vacuum system is controlled by the B&R Industrial Automation PLC model X20CP3584 [18]. The PLC software was developed in C in the B&R Automation Studio 4.5 integrated development environment. Interaction with the global control system is carried out through the Modbus TCP protocol.

Precise Temperature Regulation System

Its planned to use the Unichiller 100-H circulators from Huber [19] for precise temperature regulation. A special Pilot ONE unit would be used to control the circulators. This unit is equipped with a touch screen for local control and also supports remote control via Ethernet, which in turn makes it possible to integrate it into a global control system.

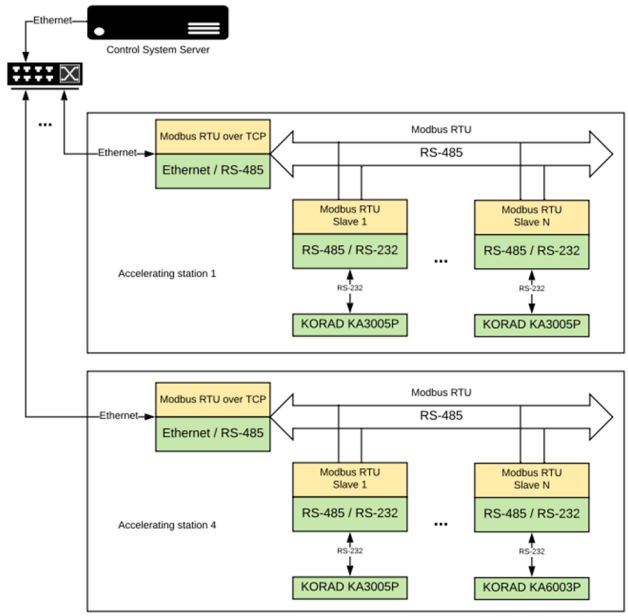


Figure 5: Magnetic element control system layout.

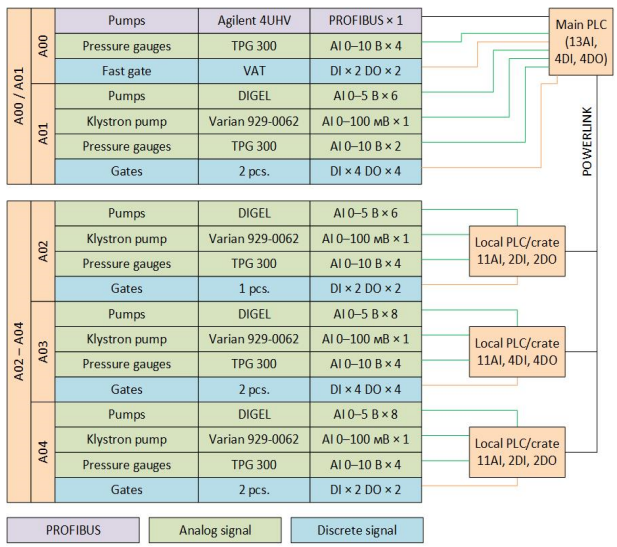


Figure 6: Vacuum control system layout.

CONCLUSION

The general concept of the global control system which capable of providing launch and control of the main accelerator subsystems has been designed. Tango-based software for individual subsystems of the Linac-200 has been developed.

The accelerator building is undergoing major repairs. After the completion of the first phase of the repair, it is planned to launch the accelerator with a new control system.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ACKNOWLEDGEMENTS

We are grateful to:

- G. Sedykh, E. Gorbachev, V. Elkin and D. Egorov for useful discussions about control system architecture and Tango-based software development advices.
- I.Shirikov, D. Donets and D. Ponkin for their valuable contributions to the Linac-200 hardware development.
- R. Pivin, A. Popov and R. Timonin for their valuable contributions to vacuum control system development.
- All Linac-200 engineers who participated in discussions and provided us with insightful comments and ideas.

REFERENCES

[1] M. A. Nozdrin *et al*., "Linac-200: A new electron test beam facility at JINR", in *Proc. 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, Brazil, May 2021, pp. 2697–2699. doi: 10.18429/JACoW-IPAC2021-WEPAB042

[2] M. A. Nozdrin, "A set of hardware-software control and diagnostic tools for the Linac-200 electron accelerator and the prototype of the JINR photoinjector," Cand. Sci. (Tech. Sci.) Dissertation, Joint Inst. Nucl. Res., Dubna, Russia 2018.

[3] M. A. Nozdrin, V. V. Kobets, R. V. Timonin, A. N. Trifonov, G. D. Shirkov, A. S. Zhemchugov, "Design of the new control system for Linac-200", *Physics of Particles and Nuclei Letters*, vol. 17, no. 4, pp. 600–603, 2020. doi.org/10.1134/S1547477120040342

[4] E.V. Gorbachev *et al*., "Nuclotron and NICA Control System Development Status", in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, October 2015, pp. 437–440. doi:10.18429/JACoW-ICALEPCS2015-MOPGF149

[5] TANGO Controls, http://www.tango-controls.org

[6] Supermicro, https://www.supermicro.com

[7] G. Strangolino, F. Asnicar, V. Forchi, C. Scafuri, "QTango: a Library for Easy Tango Based GUIs Development", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, October 2009, paper THP096, pp. 865–867.

[8] G.Sedykh, V.Elkin, E.Gorbachev, "Tango Web Access Modules and Web Clients for NICA Control System", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, October 2017, pp. 806–808. doi:10.18429/JACoW-ICALEPCS2017-TUPHA167

[9] Taurus, https://www.taurus-scada.org

[10] Weintek Labs., https://www.weintek.com/

[11] M. Nozdrin, V. Kobets, V. Minashkin, A. Trifonov, "Linac-200 Gun Control System: Status and Plans" presented at ICALEPCS'21, Shanghai, China, October 2021, paper MOPB018.

[12] Tango Socket Class, http://svn.code.sf.net/p/tango-ds/code/DeviceClasses/Communication/Socket

[13] Tango Modbus Class, http://svn.code.sf.net/p/tango-ds/code/DeviceClasses/Communication/Modbus

[14] PyModbus, https://github.com/riptideio/pymodbus

[15] E. Heine, "The MEA modulator", NIKHEF report, Nov. 12, 1998.

[16] V. V. Kobets *et al.*, "Modernization the modulators klystrons accelerating stand of the electron linear accelerator LINAK-800", *in Proc. of the 14th Russian Particle Accelerator Conference*, Obninsk, Russia, Oct. 2014, paper WEPSB02, pp. 157–158.

[17] A. Trifonov, M. Gostkin, D. Donets, V. Kobets, D. Leushin, M. Nozdrin, D. Ponkin, I.Shirikov, "Automated control system of magnetic elements for focusing and beam position correction of the LINAC-200 accelerator", *Instruments and Experimental Techniques*, vol. 64, no. 3, pp. 152–154. (in Russian). doi: 10.31857/S0032816221020270

[18] B&R Industrial Automation, https://www.br-automation.com

[19] Huber Kaltemaschinenbau AG., https://www.huber-online.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL01

DISTRIBUTED CACHING AT CLOUD SCALE WITH APACHE IGNITE FOR THE C2MON FRAMEWORK

T. Oliveira*, D. Martin Anido, M. Braeger, B. Copy, S Halastra, A. Papageorgiou Koufidis
CERN, Geneva, Switzerland

Abstract

The CERN Control and Monitoring platform (C2MON) [1] is an open-source platform for industrial controls data acquisition, monitoring, control and data publishing. Its high availability, fault tolerance and redundancy make it a perfect fit to handle the complex and critical systems present at CERN. C2MON must cope with the ever-increasing flows of data produced by the CERN technical infrastructure, such as cooling and ventilation or electrical distribution alarms, while maintaining integrity and availability. Distributed caching [2] is a common technique to dramatically increase the availability and fault tolerance of redundant systems. For C2MON we have replaced the existing legacy Terracotta [3] caching framework with Apache Ignite [4]. Ignite is an enterprise grade, distributed caching platform, with advanced cloud-native capabilities. It enables C2MON to handle high volumes of data with full transaction [5] support and makes C2MON ready to run in the cloud. This article first explains the challenges we met when integrating Apache Ignite into the C2MON framework, and then demonstrates how Ignite enhances the capabilities of a monitor and control system in an industrial controls environment.

INTRODUCTION TO C2MON

C2MON is an open-source monitoring platform developed at CERN. C2MON acts as the backbone of the Technical Infrastructure Monitoring system (TIM) that is used to monitor and control CERN's technical services from the CERN Control Centre (CCC) [6]. The main function of TIM is to provide reliable and real-time data to CCC operators about the state of CERN's widely distributed technical infrastructure. To handle such a large volume of information while maintaining data integrity, C2MON uses Java Message Service (JMS) [7] technologies together with caching technologies. Caching involves storing information in a separate low-latency data-structure for a period of time to be reused and consequently minimizing the cost of re-accessing it [2]. The existing C2MON caching layer relied on a legacy Terracotta Ehcache framework [3]. Ehcache is a widely-used Java-based cache that is fast, lightweight and can be scalable through the use of a Terracotta Server that provides distributed caching capabilities [3].

C2MON uses a 3-tier architecture, as presented in Fig. 1, that composes a Data Acquisition (DAQ) Layer, the Server Layer and a Client Layer. The DAQ layer is responsible for acquiring data from specific sources and publishing it to the C2MON server tier. The Client layer provides various service classes that allow to interact with the server. The Server layer, which is the core part of C2MON, is responsible for receiving and handling the data.

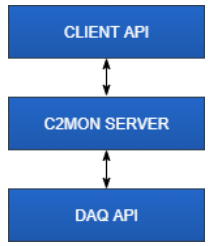


Figure 1: C2MON Architectural Overview.

Most of the information in C2MON is stored and used in the form of Tag which changes frequently as new data from the DAQ is received and evaluated by the system. In order to evaluate the correct and normal values of each Tag, C2MON provides an Alarm mechanism. The Alarm is a declaration associated with a Tag and contains a condition specifying the legal values of that Tag. If the new value received is outside the legal value range, the Alarm is activated and pushed to the client. Finally, the C2MON server layer also provides a rule engine with a set of operations, that allows expressing complex computations, comparisons and conditions.

INTRODUCTION TO APACHE IGNITE

"Apache Ignite is an open-source memory-centric distributed database, caching and computing platform" [4].

Ignite provides a simple interface to work with large data sets in real time. It is written in pure Java, based on Spring and supports different technologies like Java, C# and C++. The main capabilities that Apache Ignite provides are

- Elasticity: The Ignite cluster can grow horizontally simply by adding new nodes over a TCP connection.
- Persistence: Cache entries can be persisted on a file system or in an RDBMS (Relational Database Management System).
- Distributed computing: Apache Ignite provides a set of APIs that facilitate the distribution of computation and data processing across the nodes in the cluster for better performance; it simplifies greatly the development of a microservice-based architecture.
- Streaming: Ignite allows the processing of continuous streams of data (which C2MON uses to receive cache events asynchronously through continuous queries).

Ignite includes the notion of client and server nodes, where a node is a single Ignite instance running in a JVM (Java Virtual Machine). In a client server architecture both client and server nodes are interconnected with each other. The server, which can be constituted by a single node or a group

* tiago.marques.oliveira@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

of nodes forming a cluster, handles all the data storing processes and computing of data. The client node is usually embedded with the application code and acts as an interface to run operations in the cache like inserting or retrieving data, but can in some cases also participate in computing tasks [4]. Apache Ignite comprises a memory-centric architecture that can be used as an In-memory or distributed database. This memory-centric storage allows to store data and indexes both in-memory and in-disk with the same data structure and enables executing SQL queries in both with optimal performance.

HOW CACHING WORKS IN C2MON

In such a fast paced environment as the one present at CERN where it is crucial to maintain the correct functionality of critical systems, it is essential to have a reliable monitoring platform capable of providing high availability and fault tolerance when dealing with the increasing flows of data produced by the CERN Technical Infrastructure. In order to achieve those requirements, C2MON relies on caching technologies. The cache in C2MON acts as a working memory that allows for stored information to be retrieved as fast as possible. The database is a persistent storage, and serves as a backup solution, that is responsible for storing long term data and to populate the cache when necessary.

C2MON uses a classic client/server cache with a distributed cluster. This caching layer serves as a middleware that reduces time-consuming operations of fetching data from the database by storing the objects that are most used. When the cache starts up, it must be warmed up : cache loaders in C2MON are responsible for converting the data loaded from the database to POJOs (Plain Old Java Objects) and storing them in the cache [8].

The infrastructure of C2MON heavily relies on the use of ActiveMQ [9] to transport data. ActiveMQ is an open source, Java Messaging Service (JMS) compliant, message-oriented middleware supported by the Apache Software Foundation. The number of messages that are received can reach up to a few millions per day, and C2MON has to process all of them in real-time while maintaining data consistency. This processing involves the evaluation of Tag values, activation or termination of Alarms, rule calculation and event propagation. The calculation of rules in C2MON is a very expensive process that involves many successive calculations and can even be recursive. As the rate is so fast, some sort of rapid access memory is necessary to store intermediate values. Since JMS is only responsible of receiving and passing data and does not allow the storage of data, a new component is needed to fulfill this requirement. This logic could also be implemented directly in the database layer, as C2MON uses a relational database like Oracle. But this would require business logic to be implemented outside the C2MON server core, it would require the use of Oracle-specific Procedural Language and would lead to a lot of latency when persisting all the intermediary values. The solution is the use of a distributed working memory like Apache Ignite. This helps to achieve availability and consistency of data.

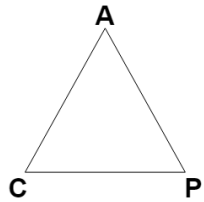


Figure 2: CAP theorem.

ADVANTAGES OF APACHE IGNITE

The Apache Ignite architecture is flexible enough to be used in various architecture patterns and styles. One possible approach that facilitates local and integration tests is to run Ignite embedded with the application where the node runs on the same JVM with the application. In a production environment a better approach is to have the Apache Ignite server nodes in a separate JVM and client nodes that connect remotely to the servers. The great advantage of this approach is the increasing flexibility that allows Ignite servers to be taken down and restarted without harming the cluster or the correct functioning of the application they support [10]. Ignite also offers the possibility of using three approaches of caching topology, where each cache mode is individually configured for each cache. In C2MON, Apache Ignite was configured with a replicated caching topology in order to achieve the highest possible level of performance. In this approach, cached data is replicated to all members of the cluster so it is available immediate use without waiting and is thus providing the fastest read-access of all caching topologies. The main downside of this approach is that writing operations are very expensive since updating a replicated cache requires pushing the new version to all cluster nodes which may limit the scalability with high frequency of updates [10].

CAP Theorem

There is a widely known idea that needs to be taken into account when dealing with distributed computing theory, which is the CAP Theorem introduced by Eric Brewer [11]. This theorem presents the fundamental trade-off between consistency, availability and partition tolerance. Figure 2 presents the graphical representation of the CAP Theorem where the three properties represent:

- Consistency that implies that all the nodes in the cluster have the same data.
- Availability of the system that should always answer the queries if possible.
- Partition Tolerance meaning that if a break in the communication occurs, the system will continue to work as expected.

Based on this theorem, Apache Ignite can be classified as a CP system, meaning that availability is sacrificed for consistency of data and partition-tolerance, since it is ACID (Atomicity, Consistency, Isolation, Durability) compliant,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL01

supporting distributed transactions with partitioned tolerance [10]. Ignite can also be considered as an AP system, since it offers two types of transactional modes for cache operations: atomic or transactional. In atomic mode Ignite supports multiple atomic operations, successively, where each DML (Data Manipulation Language) operation will either succeed or fail without any data being locked, giving it a high performance. On the other hand, in transactional mode, the DML operations can be grouped in one transaction and the data will be locked [10]. This second approach is used by Ignite in C2MON, with CP prioritised, where the consistency of data is crucial.

DIFFICULTIES AND CHALLENGES FACED

Having covered the background to this project and giving an overview of the caching system, the focus now shifts to the difficulties and challenges faced during the process of replacing the caching system.

Code Rigidity

One major problem with the design of C2MON was the rigidity of its caching layer design. Rigidity is the tendency for software to be difficult to change and where every change can cause subsequent changes in dependent modules [12]. This was due to the tight coupling of C2MON to the current caching technology, so the change would involve refactoring a major part of the current architecture.

Testing

It is crucial to have a good testing environment to ensure that the software is working as expected, and to detect and fix any eventual errors or bugs, when dealing with large and critical systems like C2MON. This should include not only unit tests to validate individual components but also integration tests to check the good behavior of those units interacting together. Integration testing environments should mimic production as much as possible, so that potential issues in production can be reproduced and resolved with certainty. This step becomes even more difficult with the increasing number of external dependencies like ActiveMQ, Oracle database and Apache Ignite, in the case of C2MON.

Cloud Readiness

Over the past years, CERN has embraced cloud technology which presents significant advantages. It allows a more agile sharing of resources and it simplifies the reusing and duplication of entire groups of machines for testing purposes. Cloud-based deployment file systems are also usually transient and network interfaces are typically allocated on the fly with a randomly-generated hardware address and attached to a local, private and non-routable network. Furthermore, the life cycle of a cloud container hosting an application is linked directly to its main process, which must consequently be managed at level of the hosting cloud, as opposed to the host they are running on [13]. C2MON was initially designed around 2009, when cloud technologies were not as ubiquitous as today, and its legacy architecture makes it impossible to take advantage of all the capabilities of a platform like Apache Ignite that is considered "cloud-ready". An example of this is the challenge node discovery in a cloud. Nodes in a cloud that wish to regroup in a cluster must usually employ broadcast messages, but this is disallowed at CERN for performance and security reasons. As a result, C2MON must rely on a fixed list of IP addresses, severely limiting how fast it can be deployed and reconfigured over new hardware or a dynamic cloud environment.

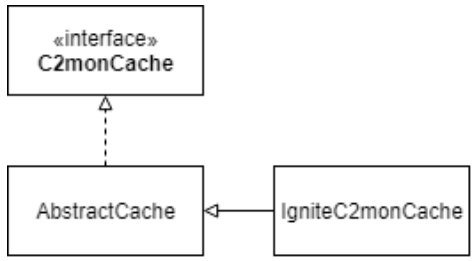


Figure 3: C2MON Cache Abstraction Class Diagram.

SOLUTIONS

The remainder of the paper will describe the redesign of the C2MON platform caching layer, and how the design choices have helped to overcome these aforementioned challenges and achieve the goal of integrating Apache Ignite into the C2MON framework.

Code Refactoring

In order to change the caching layer technology of C2MON, it was necessary to make a big code refactoring of various modules that were tightly coupled to it. To minimize the rigidity of C2MON's infrastructure, the caching layer was rebuilt depending completely on abstractions, decoupling it from the technology in place. Every dependency in the design should target an interface, or an abstract class, and never a concrete class that is much more prone to changes [12]. This approach has the disadvantage of losing some of the capabilities of Apache Ignite, but has the advantage of making it easier to change the caching technology in the future (if necessary), without major code changes. Figure 3 presents a snippet of the class diagram that corresponds to the new C2MON cache abstraction layer.

The `C2monCache` interface is the base of the entire C2MON caching layer, and contains all the necessary methods that interact with the cache, like inserting elements in the cache and retrieving them and querying elements in the cache, that was purposely implemented in a very simple way so that it could be implemented by any future alternative caching technology. This interface is implemented by an abstract class, `AbstractCache`, that hides the underlying implementation of the caching technology, and is also responsible for managing the caching update flow, the cache loaders and the cache listeners. The cache loaders are used to warm up the cache on startup and the cache listeners are used

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

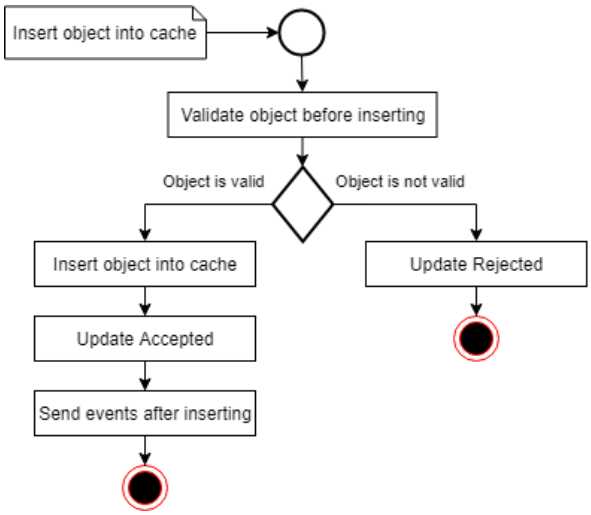


Figure 4: C2MON Cache Insertion flow.

to propagate all caching events. The `IgniteC2monCache` is the concrete class that contains the Apache Ignite implementation of `C2monCache`. Another implementation choice was the use of Java Generics throughout the whole caching implementation. Generics allows to abstract over types and the core idea is to express intent, and mark the respective objects as being restricted to contain a particular data type [14]. The `Cacheable` interface is implemented by all objects that reside in C2MON caches, creating another layer of abstraction and removing the possibility of run time errors during execution. Although Apache Ignite supports ACID transactions this might not be present in other caching technologies, and data consistency must be guaranteed for the good functioning of C2MON. Figure 4 displays an activity diagram of how the insertion in the cache is processed to respect cache consistency. When a new object is inserted in the cache it goes through a validation process first that guarantees that the object is complete and correct, and it is not older than the previous update, otherwise the update operation is discarded. In case the validation passes, the object is successfully inserted in the cache and corresponding events are propagated to any potential cache listeners.

Testing

The lack of a proper testing environment, made it difficult to fully test the correct integration of C2MON with its various external dependencies. The alternative used to overcome this problem was the use of Testcontainers [15]. Testcontainers is a Java library that supports JUnit tests and that can execute those tests against instances of any software that can run in a Docker container. To use Testcontainers, all of C2MON components had to be ”containerized” for the Docker container platform. Testcontainers encapsulate the management of the Docker images and spins up the required containers during the tests and tears them down once the test execution is finished. It provides a faster feedback loop for integration tests, as all the containers start with a clean and known state which increases the reliability of the tests.

CONCLUSION

The usage of Apache Ignite has proven to be a robust and capable solution, that enhanced the capabilities of the C2MON framework. However, some of its capabilities could not be completely exploited at CERN because of the way C2MON is deployed. Although the main goal was to change the caching technology of C2MON and to integrate Apache Ignite, the opportunity was also taken to sanitize and refactor the entire caching layer. This was done following the best practices for software development and testing, employing newer features like the implementation of generics and the use of Java streams. Product testability proved to also be a very important factor that had to be improved to ensure that C2MON behaved the same way as before the Ignite integration : Docker technology (through Testcontainers) proved an effective and simple way to perform extensive, repeatable integration testing of the C2MON platform.

REFERENCES

[1] M. Braeger *et al.*, “A customizable platform for high-availability monitoring, control and data distribution at CERN”, in *Proc. ICALEPCS 2011*, Grenoble, France, 2013, pp. 418–421.

[2] Greg Luck and Brian Oliver, “Java™ Caching API - The Java Caching API is an API for interacting with caching systems from Java programs”, Dec. 2013. `https://download.oracle.com/otn-pub/jcp/jcache-1_0-fr-spec/JSR107FinalSpecification.pdf`

[3] Software AG, “About Terracotta Ehcache”. Apr. 2019. `http://documentation.softwareag.com/onlinehelp/Rohan/tc-ehcache_10-3/10-3_About_Ehcache.pdf`

[4] Shamim Bhuiyan and Michael Zheludkov, “The Apache Ignite Book - the next phase of the distributed systems”, 2019.

[5] Paul Parkinson, “Java™ Transaction API (JTA) - Version 1.2”, May 2013.

[6] J. Stowisek, A. Suwalska, and T. Riesco, “Technical infrastructure monitoring at CERN”, in *Proc. EPAC’06*, Edinburgh, Scotland, 2006, paper TUPLS135, pp. 1822-1824.

[7] Richard Monson-Haefel and D. Chappell, “Java™ Message Service (JMS)”, 2000.

[8] Szymon Halastra, “Refactoring of the CERN in-memory Data grid”, 2019.

[9] Bruce Snyder, Dejan Bosanac, and Rob Davies, “Introduction to Apache ActiveMQ”, August 2008.

[10] Shamim Bhuiyan, Michael Zheludkov, and Timur Isachenko, “High Performance In-Memory Computing with Apache Ignite - building low latency, near real-time application”, 2017.

[11] Seth Gilbert and Nancy A. Lynch, “Perspectives on the CAP Theorem”.

[12] R. C. Martin, “Design Principles and Design Patterns”, 2000.

[13] B. Copy, M. Braeger, E. Mandilara, F. Ehm, and A. Lossent, “C2MON Scada Deployment on Cern Cloud Infraestructure”, 2017.

[14] Gilad Bracha “Generics in the Java Programming Language”, January 2004.

[15] Testcontainers, `https://www.testcontainers.org/`.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

IMPLEMENTING AN EVENT TRACING SOLUTION WITH CONSISTENTLY FORMATTED LOGS FOR THE SKA TELESCOPE CONTROL SYSTEM

S.N. Twum*, W. Bode, A. F. Joubert, K. Madisa, P.S. Swart, A. J. Venter
SARAO, Cape Town, South Africa
A. Bridger, UKATC, Edinburgh; SKAO, Macclesfield

Abstract

The SKA telescope control system comprises several devices working on different hierarchies on different sites to provide a running observatory. The importance of logs, whether in its simplest form or correlated, in this system as well as any other distributed system is critical to fault finding and bug tracing. The SKA logging system will collect logs produced by numerous networked kubernetes deployments of devices and processes running a combination off-the-shelf, derived and bespoke software. The many moving parts of this complex system are delivered and maintained by different agile teams on multiple SKA Agile Release Trains. To facilitate an orderly and correlated generation of events in the running telescope, we implement a logging architecture which enforces consistently formatted logs with event tracing capability. We discuss the details of the architecture design and implementation, ending off with the limitations of the tracing solution in the context of a multiprocessing environment

PREVIEW TO THEORY AND SKA SYSTEM ARCHITECTURE

Observability and Monitoring in a Distributed System

Logs have long been the de facto approach used to sample parts and peek into the internal state of a running program. Coupled with metrics, monitoring can be done across a system to understand its health at any given time. Inasmuch as this age old approach has been very beneficial to developers, especially for debugging purposes, it is limited in its diagnostic ability in distributed environments. Monolithic applications are easily observable using only logs and metrics. But in the dawn of the era of microservice architecture, logging alone is not adequate to debug and probe the internal state of such a system. Distributed systems with different services and multiple instances of these services need a correlated view of events to troubleshoot errors. It now requires the use of logs, metrics and traces, all together producing the emergent quality of this new buzz word, "observability". J. Heather explains observability as inferring the internal state of a system from its external outputs. But it is not just the ability to see what is going on in your systems. It's the ability to make sense of it all, to gather and analyze the information you need to prevent incidents from happening, and to trace their path when they do happen, despite every safeguard, to make sure they don't happen again[1]. Full observability of a distributed system is a function of three pillars [2], namely:

- Logs: a snapshot of an event in a running system.
- Metrics: measurement of activities on a running system, e.g. CPU load.
- Tracing: A trace is a representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system [2].

An Overview of the SKA Telescope Control System Architecture

The SKA telescope control system will be a collection of software and services running from two sites which will be controlled from HQ in Jodrell Bank. The software consists of Tango Devices managing specific telescope hardware, and processes running all manner of software which are maintained by 17 or more teams on our Agile Release Trains (ARTs) [3]. The control system has a hierarchical structure to reduce complexity. The TANGO device is an abstraction hierarchy that has functional purpose at the top, and that goes down to physical form at the bottom. The frequency of intervention required at the different levels are different, increasing as you go downward the hierarchy (the Telescope Operator will exercise low frequency supervisory control, while at the lowest level you find real-time, closed loop feedback control) [4]. Control propagates downward (with fan-out) through this hierarchical structure. This is one source of causal path for events to propagate through the system.A specialisation of this hierarchical structure is that of the sub-array, an aggregation of telescope resources that are engaged for use in an observation [5]. Figure 1 illustrates the usage of sub-array nodes to control of the Mid telescopes.

During the lifetime of a running telescope, commands are triggered, events are fired, threads are spawned, several things are happening at the same time and it is not a trivial task to have end to end observability of the running system. This complex network exudes all the characteristics and challenges that are inherent in distributed systems. Though the various moving parts of the system are well tested, they are susceptible to faults and the ability to pin point a glitch to an exact root cause, following it through the various parts can only be provided in a distributed tracing system.

* stwum@sarao.ac.za

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

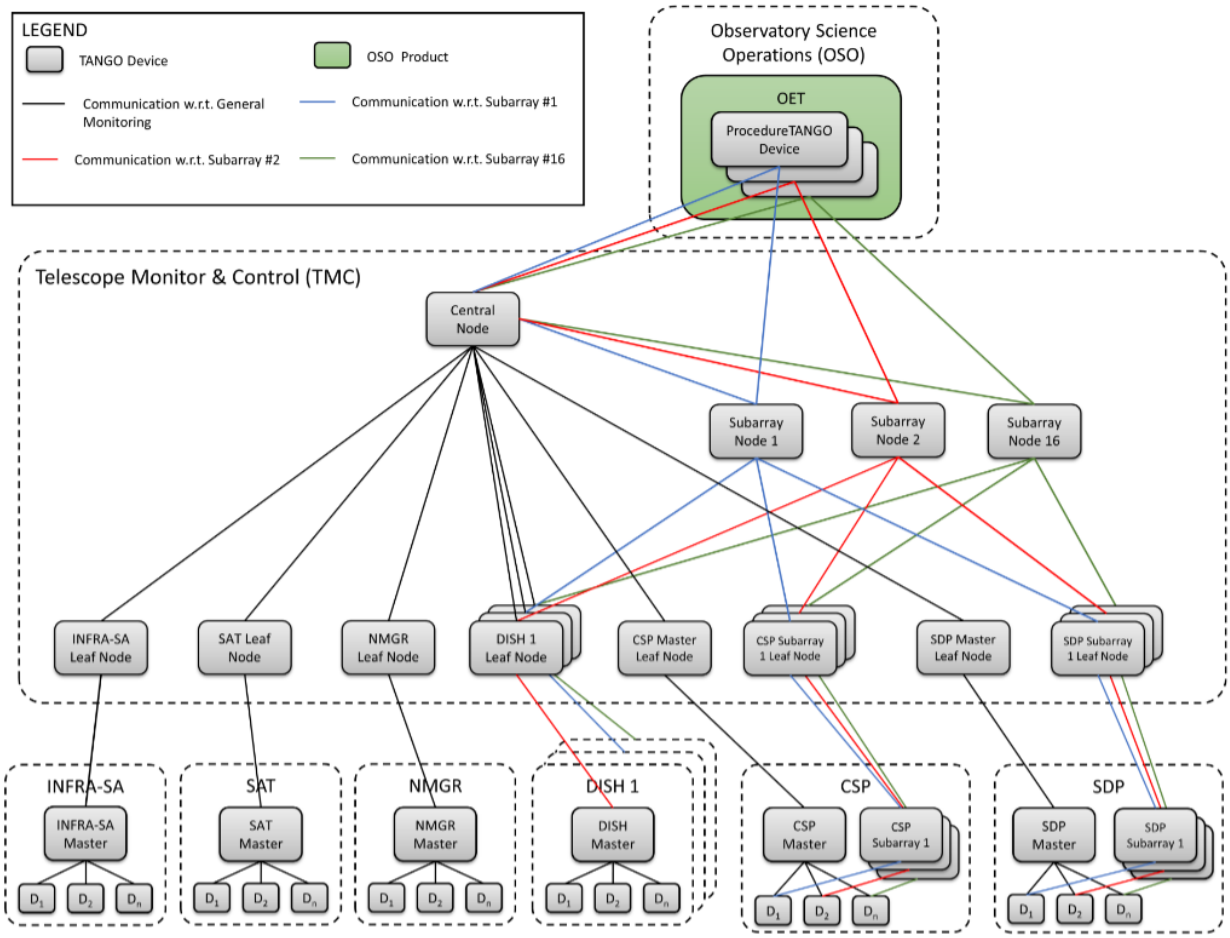


Figure 1: Hierarchy of tango devices demonstrating the control of the Mid telescopes using sub-array nodes [5].

In order to glean correlated debugging information for the SKA distributed services/devices architecture, we first adopted a standardised approach to logging across all packages maintained by the different teams and then, implemented a tracing solution in the form of transaction IDs to be used by Tango commands. The remaining sections discuss in detail the design decisions on our logging standard and the transaction IDs used to provision our tracing solution.

HARMONISING APPROACHES TO LOGGING IN THE SKA SERVICES/DEVICES SYSTEM

The SKA logging system collects, stores and retrieves logs from derived, off-the-shelf and bespoke software. All these different logs sources are maintained in different packages by different teams on the SKA Agile Release Train. Prior to the harmonised logs, each team had their own logging standard. After a review of the different logging approaches used by the teams, we provided a reference logging implementation which implements a standard log format. The details about the standard log format follow in the next section.

SKA Standard Log Format

The SKA log format ensure logs are emitted in a uniform manner to aid in troubleshooting and parsing. All processes executed inside a container log to `stdout`. For the emitted logs to be ingested into the logging system, they have to conform to the format below [6]:

```
VERSION"|"TIMESTAMP"|"SEVERITY"|"[THREAD-ID]"|
"[FUNCTION]"|"[LINE-LOC]"|"[TAGS]"|"MESSAGE LF
```

Example log in the ska log format is below:

```
1|2019-12-31T23:12:37.526Z|INFO||testpackage.
testmodule.TestDevice.test_fn|test.py#1|
tango-device:my/dev/name| Regular information
should be logged like this FYI
```

The log message format is not an extension of syslog/RFC5234 format. More on the SKA log message format can be found on the SKA Telescope Developer Portal [1].

To aid in filtering, Table 1 below describes tags that have been adopted as standard tags which may be found in a log message.

[1] https://developer.skatelescope.org/en/latest/tools/logging-format.html

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

Table 1: SKA Standard Tags for Logs [6]

Tag Name	Description	Example
tango-device	An identifier string in the form:<facility> /<family> /<device>. This corresponds with a Tango device name. • facility: The TANGO facility encodes the specific telescope (LOW or MID) and the telescope sub-system (SaDT, TM, SDP, CSP, Dish, LFAA, INFRA) • family: Family within facility • device: TANGO device name	MID-D0125/rx/controller • MID-D0125: where 0125 is the serial number of the Dish instance • rx: The Single Pixel Feed Receiver of the Dish • controller: The controller of the Single Pixel Feed Receiver
subsystem	For software that are not TANGO devices, the name of the telescope sub-system.	SDP
transaction_id	Transaction ID to associate logs from different systems relating to a distributed activity	txn-t0001-20200928-000000010

Design Motivation

The preliminary results gathered from analysing the code base of existing projects at the time drove the design of our current standard. We prioritised:

- Readability
- A sensible size of log length to provide useful information. The minimum required are:
  - timestamp
  - log level
  - extensible tags - a mechanism to specify arbitrary tags
  - the function in the source code where the log emanates from
  - the filename where the log call originates
  - the line number in the file
- Ability to parse the log message.

The SKA Logging Configuration Library

The SKA logging format is hosted in the ska-telescope gitlab repository as a python package. The package allows developers to configure logging for any module to have all applications log in a consistent format [7]. The SKA tags defined in Table 1 above can be added to a tags filter available in the library among other configurations. All new ska projects in both the Observation and Management Control and Data Processing ARTs use this logging library as do the existing ones before the format and its library were implemented.

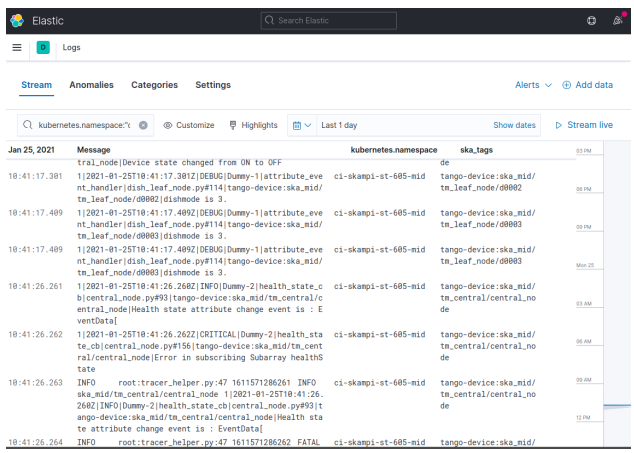


Figure 2: Kibana logs filtered using the value of the ska_tags_field.tango-device field [6].

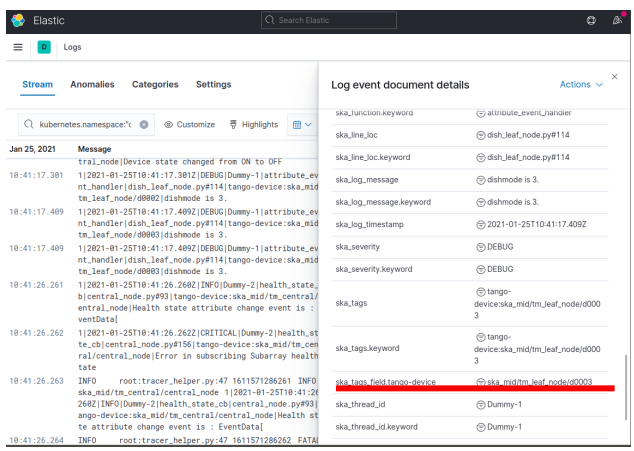


Figure 3: Kibana logs with only messages with the value ska_mid/tm_leaf_node/d0003 for the ska_tags_field.tango-device field [6].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

TRACING EVENTS IN THE SKA CONTROL SYSTEM USING TRANSACTION IDS

The SKA control system churns out logs and aggregates metrics for system health monitoring; the missing piece to the full observability puzzle was the ability to obtain correlated log trails (tracing) of events. Tracing is the the only signal which explains the relationship between the different parts of the system which are managed by the different teams in our ARTs. It is nearly impossible to debug and discover patterns or correlations in the SKA control system searching through the gigantic logs being aggregated from the many services running in the kubernetes orchestrated environment. With the SKA formatted logs complete, it served as a scaffold to build the tracing solution by injecting a transaction ID in the logs. This design was driven by the snippet below:

```
with transaction('My Command') as txn_id:
    # do stuff
    ...
```

The idea in the snippet above is to ensure that any action performed within the context of the transaction will have record of an id for tracing. The transaction ID is cascaded down further requests until a trace is complete. In the SKA control system, the transaction context handler is implemented in a transactions python library [2] which relies on the SKA Logging [3] and SKA Unique Identifier (SKUID) [4] libraries.

SKA Unique Identifier Library

The SKUID library does more than just generate unique ids for transactions. It generates scan ids, entity ids and transaction IDs for telescope operations [8]. The scan and entity ids are out of this discussion's context. The transaction ID is generated every time a transaction context block is initialised. It is a unique value served by a remote or local unique id generator (which can possibly generate a duplicate). The remote URL has a `generator-id` following right after the "`txn`" prefix (e.g. `txn-t0001-20200914-123456789`) while the local generator has "`local`" right after the "`txn`" prefix (e.g. `txn-local-20200921-516590971`).

SKA Log Transactions Library

This library glues together the SKUID and logging packages to provide a transaction ID in a context handler which is injected into logs as a tag [9]. Within the context handler there is an enter and exit log entry which log the beginning and the end of a transaction using the generated transaction ID or a specified custom ID. In the event an exception occurs within the transaction, an exception log will be emitted with the transaction ID also. An example log on entry and exit of the transaction context in the ska log format looks like this:

```
1|2020-10-01T12:49:31.119Z|INFO|Thread-210|
log_entry|transactions.py#154||Transaction
[txn-local-20201001-981667980]: Enter[Command]
with parameters [{}] marker[52764]

1|2020-10-01T12:49:31.129Z|INFO|Thread-210
|log_exit|transactions.py#154||
Transaction[txn-local-20201001-981667980]:
Exit[Command] marker[52764]
```

To demonstrate the usage of the log-transaction library, a multi-level device has been added as an example in the ska-tango-examples [5] library. This multi-level device compromises a top-level device, four mid-level devices and one low-level device as depicted in Fig. 4.

The mid-level devices are triggered from commands in the top-level device and they in turn call the low-level device. During the interaction between these devices, the transaction ID generated at the top-level is propagated all the way to the low-level device and appears as tags in the logs as shown in Fig. 4. The rest of the logs are a combination of logs showing entry and exit into mid-level and low-level devices. When all mid-level devices with their associated low-level device triggers are finished, the trace ends with a log indicating an exit from the top-level device [10].

Tracing in a Multithreaded Environment

Though the log transactions library has support for async code it does not support a multithreaded case at the moment. Consider the scenarios in the code snippets below:

```
# SCENARIO 1
class Device(Device):

    def function_that_logs(self):
        do_something()
        log.info("logging something") # The
        ↪ same thread, so this should log
        ↪ a transaction ID

    @command
    def some_command(self):
        with transaction("name", parameters,
        ↪ logger=ska_logger) as
        ↪ transaction_id:
            function_that_logs()
```

[2] https://gitlab.com/ska-telescope/ska-ser-ska-ser-log-transactions
[3] https://gitlab.com/ska-telescope/ska-ser-logging
[4] https://gitlab.com/ska-telescope/ska-ser-skuid

[5] A project that demonstrates how to structure an SKA project that provides some simple Tango devices coded in PyTango. See https://gitlab.com/ska-telescope/ska-tango-examples

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

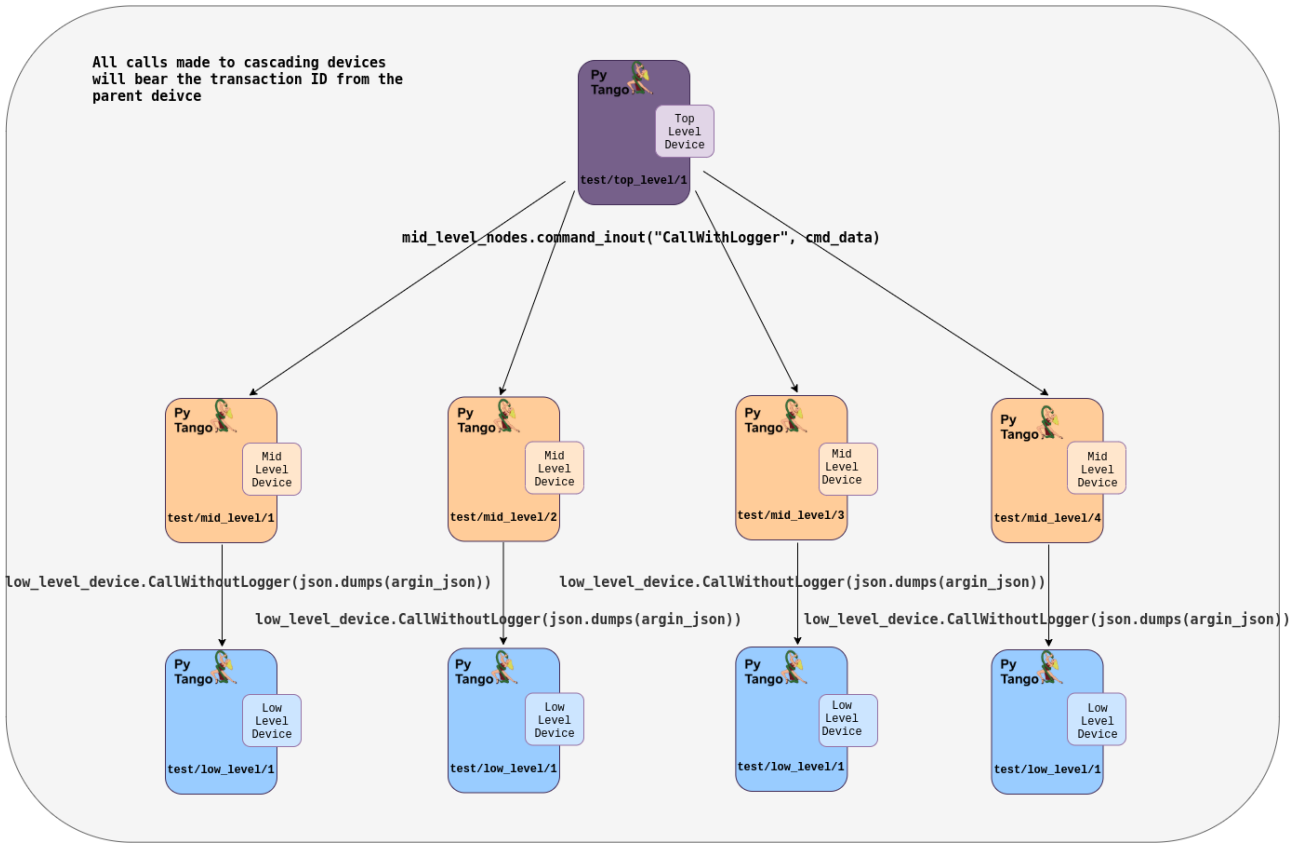


Figure 4: Diagrammatic illustration of the tango-example multi-level device and interaction among those devices.

```
 # SCENARIO 2
class Device(Device):

    def function_that_logs(self):
        do_something()
        log.info("logging something") #
         ↪ This will not have a
         ↪ transaction ID

    @command
    def some_command(self):
        with transaction("name", parameters,
         ↪ logger=ska_logger) as
         ↪ transaction_id:
            t = threading.Thread(
                target=function_that_logs,
                 ↪ args=(1,))
            t.start()
            t.join()
```

The code in SCENARIO 1 will produce a log just like in the example log above with the transaction ID in it but the log from SCENARIO 2 will not. The same is true for work passed off from a transaction to a thread managed by a thread pool. The thread lifetimes would exceed the transaction. This applies to code with or without asyncio or gevent.

FUTURE WORK

The enablers for the functioning of the SKA log transaction, namely SKA logging and SKA unique ID generator, do not have any left over features to be added except the local ID generation of the SKUID service to guarantee absolute uniqueness. This is trivial at the moment as the SKUID service URL is served up all the time in the integration environment. Having had all three pillars of distributed system implemented, the logical next step is to build the tooling we need to facilitate system diagnosis. The tooling will be available in all ART and will display to a user the flow of information and state across the MVP. At the time of implementing the log transaction, the SKA Telescope Control System is currently been demonstrated in an MVP and there is no clear evidence of that our troubleshooting would benefit from this feature. The need for it will be assessed and implemented as the project progresses, especially with AA0.5 release at hand.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONCLUSION

To fully understand a distributed system requires us to have distributed tracing. Tracing compares to a database join with the contributing tables being the different paths the event travels through and each event having an ID on which the join is performed [11]. The SKA Control System like every other distributed system poses a big observability challenge. To be able to infer the internal state of both low and mid telescope control systems, we have developed libraries to ensure running applications have consistently formatted logs with tracing. All the work done are available in public repositories under the ska-telescope organisation in: ska-ser-logging, ska-ser-skuid and ska-ser-log-transactions.

ACKNOWLEDGEMENTS

We would like to acknowledge the National Research Foundation (NRF), the South African Radio Astronomy Observatory and the SKA Observatory for their support in carrying out this work under the Bridging Program and the SKA SAFe program.

REFERENCES

[1] J. Heather, *The New Stack, Cloud Native Observability for DevOps Teams*. Alex Williams, 2021.

[2] C. Sridharan, *Distributed Systems Observability: A Guide to Building Robust Systems*. O'Reilly Media, Inc., 2018.

[3] M. Bartolini, L. Brederode, M. Deegan, M. Miccolis, N. Rees and J. Santander-Vela, "Scaling Agile for the Square Kilometre Array", in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, https://doi.org/10.18429/JACoW-ICALEPCS2019-WEPHA011, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1079–1083, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-WEPHA011. https://jacow.org/icalepcs2019/papers/wepha011.pdf

[4] W. Findeisen, "Hierarchical control structures", *Control and Cybernetics*, 2000.

[5] P. Dewdney, "SKA1 design baseline description", Rep. SKA-TEL-SKO-0001075, Internal SKA document, 2000.

[6] SKA log message format, https://developer.skatelescope.org/en/latest/tools/logging-format.html

[7] SKA logging, https://developer.skao.int/projects/ska-ser-logging/en/latest/?badge=latest

[8] SKA unique identifiers, https://developer.skatelescope.org/projects/ska-ser-skuid/en/latest/?badge=latest

[9] SKA transaction logging, https://developer.skao.int/projects/ska-ser-log-transactions/en/latest/?badge=latest

[10] Multi level device in tango example, https://gitlab.com/ska-telescope/ska-tango-examples/-/merge_requests/28

[11] Why distributed tracing will replace (most) logging, https://www.youtube.com/watch?v=Hv98hU3nj0U

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL03

Tango CONTROLS RFCs

V. Hardion, MAXIV Sweden, Lund, Sweden
A. Götz, R. Bourtembourg, ESRF, Grenoble, France
S.Blanch-Torné, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
P.P. Goryl, M. Liszcz, S2Innovation, Kraków, Poland

Abstract

In 2019, the Tango Controls Collaboration decided to write down a formal specification of the existing Tango Controls protocol as Requests For Comments (RFC). The work resulted in a Markdown-formatted specification rendered in HTML and PDF on Readthedocs.io. The specification is already used as a reference during Tango Controls source code maintenance and for prototyping a new implementation. All collaborating institutes and several companies were involved in the work. In addition to providing the reference, the effort brought the Community more value: review and clarification of concepts and their implementation in the core libraries in C++, Java and Python. This paper summarizes the results, provides technical and organizational details about writing the RFCs for the existing protocol and presents the impact and benefits on future maintenance and development of Tango Controls.

INTRODUCTION

The TANGO control system is a device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA systems.

The first version of the Tango Controls was designed and developed more than 20 years ago [1]. Since then, it has evolved to follow technology progress and needs of new features and improve its quality. There are still some technical challenges that relate to the legacy of the source code and dependency.

One of the challenges is concerning the heart of TANGO, which use CORBA for all the client/server communication. This open architecture allows distributed objects to communicate with each other, which is a perfect match for a scientific and heterogeneous control system like Tango, considering each hardware as an object.

From its date of creation, CORBA has seen different support from big names in the industry, even being part of the standard library of programming languages like Java. This architecture was the seed of many other types of architecture like web services (JBOSS).

Motivation

Nowadays, CORBA is still used by very specific domains but not maintained to face the evolution of computer science. In 2013 the Tango Community mentioned for the first time a study for the replacement of CORBA.

The implementation reference of Tango written in C++ shows a real entanglement inside the source code, in which OmniORB, the C++ CORBA library, is leaking in all the public API. Removing CORBA would mean refactoring the entire Tango C++ library by abstracting at many different levels.

Backward compatibility in Tango is essential, and the idea to replace the library would generate a lot of uncertainty regarding the fundamentals of Tango. Having a precise specification, an idea that emerged during the 2018 Tango kernel meeting in Krakow and lacking before the RFC project, would have been the best way to remove this risk.

Implementation Agnostic The Community works towards improving the maintainability of Tango Controls. The Collaboration proactively makes the framework immune to obsolesce of libraries and technologies (e.g. CORBA) it is based on. It is expected it may require rewriting all source code-base for a particular language.

Knowledge sharing Another factor to consider is the retirement of the initial Tango Developers. There is a risk of losing a deep knowledge of the Tango protocol implementation. Before the RFCs, the tango library source code was, in fact, the only specification. This could lead to losing the compatibility between versions of the Tango Controls or between libraries for different programming languages after bug fixes or features' implementations.

Compatibility The Community has found that formal documentation of the protocol and Tango Controls concepts is needed [2]. This assures that maintenance and development will not break compatibility.

WRITING RFCS

Writing proper specifications, understandable by software developers and not influenced by the implementation, was not an exercise that the Tango community used to do.

The model of the specification was inspired by the process established by ZMQ [3] which is one of the protocols used by Tango with an open specification and very well documented process called Request For Comments (RFC).

After a presentation at the 2019 Tango Meeting (DESY, Hamburg 2019), it was decided to start writing the specification with the involvement of all major institutes and companies that use Tango Controls.

The Process

The Tango RFC process is a clone of the ZMQ RFC one with a minor adaption to the Tango organisation. The scope

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

of this RFC was to collect the existing specifications of the Tango V9 in regards to the concepts and model, protocol behaviour and conventions.

The objective is to describe the expected behaviour when an operator interacts with the model of Tango and also to describe the communication and processes which allow two different implementations of Tango to communicate between each other. Contrary to ZMQ, the API is not part of the scope of specification for two reasons. First, the current C++ implementation uses CORBA object, which would make the specification very tight to the implementation. Secondly, the different implementations use specifics of the language implemented with, i.e. High-Level Device Servers, although they are trying hard to respect a similar API.

In the future, it would be interesting to fix the API into specification. But for practicalities, it was better to focus on the abstraction formed by the Tango Model and to test this new way of working.

The process of writing the specification is based on C4 and COSS, which are very similar to how the software developers work in a git environment.

C4 which stand for Collective Code Construction Contract [4] is inspired by the Github-flow [5] but applied to documentation. This way of working ensures of the transparency of the edition by keeping track of:

- changes expected by the authors to the existing specification (logging issue),
- the different phases of building the new specification,
- the comments and the acceptance of the editors

This process allows anyone of the Tango Community to participate to the edition whatever the degree of membership. Only the editors roles was restricted to the representative of the Tango Consortium [6] to ensure a review from an experienced person.

COSS stands for Consensus-Oriented Specification System [7] is a lightweight process which describe the lifecycle of the specification. It aims at achieving small specification which are not necessary fully complete although accepted as the most useful description of the moment. Consensus-oriented means that all the participant have to find a common ground before to be validated. Further detailed specification can be achieved by iteration.

In the case of the Tango community, the knowledge has been spread over tens of different institutes which often use Tango their own way with their own semantic. COSS is important for the Tango Specification as it concentrates the point of view of many stakeholders by consensus. The lifecycle also describes what part of the specification is obsolete, also by consensus. The Tango v10 project will probably introduce deprecated feature which should be clearly stated. The complete set of state described in the COSS specification allows to cover the all lifecycle (Fig. 1).

Tools

Creating a sustainable technical specification requires good tooling support. The Tango Control RFCs are the result of a collective effort of numerous contributors. The authoring process requires collaboration, involves many discussions, and ends with a peer review of the resulting document. These factors led to the adoption of the *Docs as Code* [8] philosophy and raised a need for a full-fledged DevOps platform that facilitates planning and tracking the work, provides a version control system, allows for code reviews and enables integration with external services to publish the documents. GitHub was selected as a platform offering all the required features. After migration in 2021 [9], all RFC development takes place on GitLab [10]. The RFC documents are published on the Read the Docs platform [11].

The RFC specifications are drafted using the Markdown markup language [12]. Markdown format was selected because of its close integration with the GitHub (and now GitLab) platform. The documents are stored in human-readable plain text files which makes them suitable for putting in a version control system. GitLab automatically renders Markdown as HTML which allows one to preview the modification during the development and provides instant feedback on how will the final document look like. The RFCs are taking advantage of the extensions offered by GitLab Flavored Markdown, including the *front matter* [13] which is used for embedding metadata like editor's name or document's status.

Some aspects of the Tango Controls specification like naming schemas must describe complex textual patterns. A well-defined formal notation is required to precisely describe the structure of such patterns. The RFC authors adopted the ABNF language [14] for this purpose. Listing 1 shows an excerpt from the Device Property specification using ABNF to specify the Property name.

Listing 1: Device Property name specification.

```
alphanum = ALPHA / DIGIT
underscore = %x5F
device_property_name =
    1*1ALPHA
    0*254(alphanum / underscore)
```

THE RESULT

Documents' Structure and Organisation

The project provides a template to prepare an RFC about a given part of the specification. Each topic to specify has its own document. Always starts with some metadata to help automatic processing followed by a short preamble that focus on the licensing of this documentation and the use of the ietf RFC2113. It is important then to collect the goals of the document and some user cases to describe what it is being to be specified.

The specification itself is the most agnostic as possible to the current implementations. The focus stay in describe

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL03

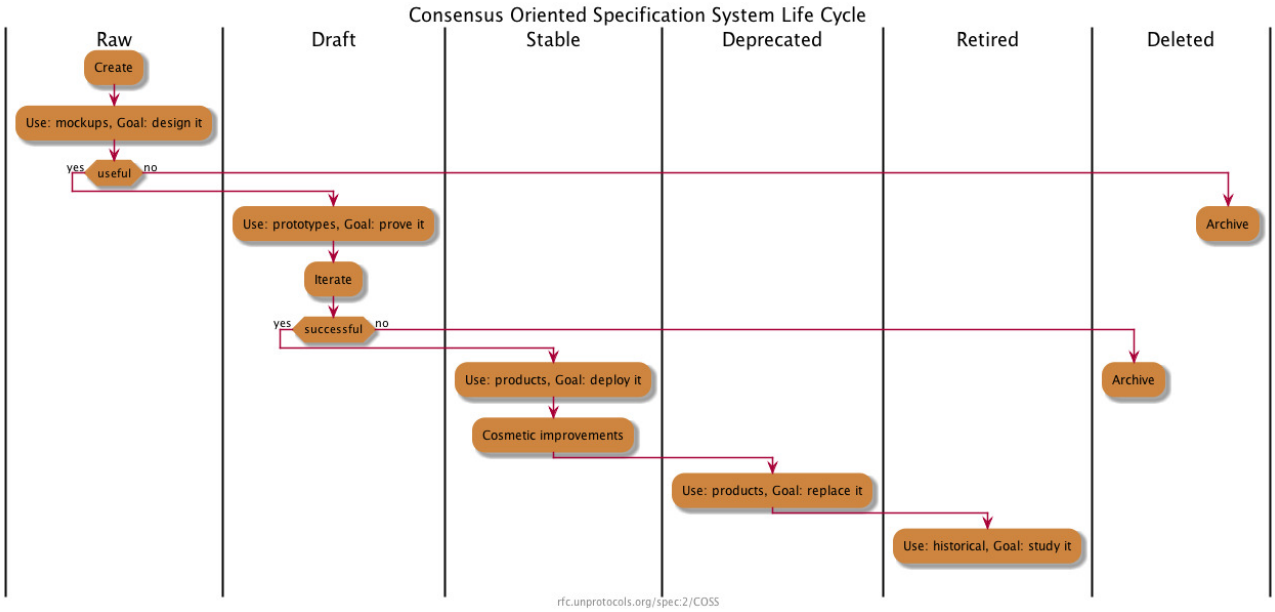


Figure 1: The complete COSS lifecycle.

and define the best way possible how things shall be made for interoperability and compatibility but giving enough freedom to avoid to be a rigid corset.

The topics in the tango-rfc are [15]:

1.Tango Control System	10.Request-Reply protocol
2.Device Object model	11.Request-Reply with CORBA
3.Command model	12.Publisher-Subscriber protocol
4.Attribute model	13.Publisher-Subscriber with ∅MQ
5.Property model	14.Logging service
6.Database system	15.Dynamic Attrs & Cmds
7.Pipe model	16.Cache system
8.Server model	17.Memorized Attributes
9.Data types	18.Authorization system

Publication

The RFCs are published on the ReadTheDocs.io service: `https://tango-controls.readthedocs.io/projects/rfc/en/latest/`. It is compiled from the MarkDown sources with use of *Sphinx* [16] and *myst-pareser* [17] library and renderd to HTML and PDF formats.

The ReadtTheDocs supports versioning. The service can be configured to use Git tags and branches as sources for different versions of the documentation. This feature allows to relate the RFCs to a specific version of Tango Controls. The versioning is already exploited to pre-release RFC-10 *Request-Reply protocol* specification [18] which is well advanced in writing but still in-progress.

USE OF THE RFCS

Added Value

Writing formal specifications for the existing Tango 9 software had some interesting side benefits.

Trying to describe some Tango 9 features revealed some inconsistencies in the way they are currently implemented in the different languages. For instance, some code generated by POGO [19], the Tango code generator, is not always behaving the same way when generated in the different supported languages. The possibility to override a device property with a class property is currently not implemented the same way in the different languages [20]. This will be improved in a future version of POGO.

This exercise helped to spread the knowledge among the involved developers on some not so well known Tango features and also pointed out the parts of the documentation which required some improvements.

This work triggered as well interesting discussions to agree on a common vocabulary because different words were used in different institutes to name the same Tango concepts. For instance, the administration device is sometimes named “Management Device”, “Admin Device”, “DServer” or “Device Server” in different institutes [21].

Finally, this activity was a great opportunity to discuss new features [22], backwards compatibility of the future versions and to propose some improvements at the concepts level.

Future Use

The RFC will play an essential role in the future development of a new major version of Tango. The model of Tango has proven to be a powerful concept for the last 20 years. Many existing and new projects have successfully

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

based their control system Tango and therefore need to ensure that new versions of Tango provide the same or compatible features. The RFCs describe the behaviour of Tango as it is today seen by clients and servers. The RFCs open the door to a new development of the Tango core with a new code base while ensuring that the features of Tango which are provided by the current LTS version (V9) are maintained. The RFCs can be seen as a specification for future versions of Tango. Having a clear specification agreed by all means new implementations are not required to implement all features of a full Tango-based system. Possible implementations could be made for only the server part or even micro-servers which implement only events for example. This approach allows a step-by-step approach to testing new ideas without taking a large risk that the new developments do not satisfy the needs of a typical Tango system.

The RFCs are also a guarantee that even if key members of the kernel team leave the behaviour of Tango has been captured and documented. This gives another guarantee that in the future of Tango could be entirely redeveloped without taking too much risk for existing installations even if source compatibility is not guaranteed.

CONCLUSIONS

The most important Tango Controls concepts and protocol behaviour are formally documented now. The RFCs are published on ReadTheDocs: `https://tango-controls.readthedocs.io/projects/rfc/en/latest/`. These documents have status *draft*. This means one can use them as a specification for prototype development.

Final work is in progress for RFC-10/The Request-Reply Protocol [18]. The RFC-13/Publisher-Subscriber protocol implementation with ZMQ is waiting for review [23]. Three other RFCs:

- RFC-11/The Request-Reply protocol - CORBA implementation,
- RFC-17/Memorised attribute service,
- RFC-18/Authorisation system

are planned to be written if need.

Cache system specification is now part of RFC-10/Request-Reply, and it is not yet decided if it will be moved to RFC-16.

The Tango Community plans to organise a workshop for the final edition of the RFCs and plan the next steps.

The RFCs will be further developed to follow new or changed features of Tango Controls. As the Tango Community plans to build a prototype of Tango V10, the RFCs will be used and verified for completeness and any ambiguity. This will also trigger either supplementation or corrections of the specifications.

Following COSS, after verification by usage and when a new version of Tango Controls is ready, the documents will be marked as stable.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Tango Controls Collaboration for funding and contributions. The RFCs were written with the engagement of all Tango Controls Community. Authors would like to thank all who contribute by writing, commenting, reviewing, discussing and encouraging to do the work, among others (in last name alphabetical order): Gwenaelle Abeillé (SOLEIL), Reynald Bourtembourg (ESRF), Thomas Braun (byte physics), Antoine Dupre (MAX-IV), David Erb (MAX-IV), Andrew Götz (ESRF), Igor Khokhriakov (IK), Olga Merkulova (IK), Sergi Rubio (ALBA), Graziano Scalamera (Elettra).

REFERENCES

[1] Chaize J.-M., Götz A., Klotz W.-D., Meyer J., Perez M., Taurel E., "TANGO - An Object Oriented Control System Based on CORBA", in *Proc. ICALEPCS 1999*, Trieste, Italy, Oct. 1999, paper WAI2I01.

[2] P.P. Goryl, A. Götz, V.H. Hardion, M. Liszcz, and L. Pivetta, "Towards Specification of Tango V10", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 331–333. doi:10.18429/JACoW-ICALEPCS2019-MOPHA051

[3] ZeroMQ, `https://rfc.zeromq.org/`.

[4] `https://github.com/unprotocols/rfc/blob/master/1/README.md`

[5] `https://guides.github.com/introduction/flow/`.

[6] A. Götz, G. Abeillé, S. Brockhauser, M.O. Cernaianu, J.M. Chaize, T.M. Coutinho, *et al.*, "The TANGO Controls Collaboration in 2015", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, paper WEA3O01, pp. 585–588, `http://jacow.org/icalepcs2015/papers/wea3o01.pdf`, `doi:10.18429/JACoW-ICALEPCS2015-WEA3O01`, 2015.

[7] `https://github.com/unprotocols/rfc/blob/master/2/README.md`

[8] `https://www.writethedocs.org/guide/docs-as-code/`.

[9] M. Liszcz *et al.*, "Migration of Tango Controls Source Code Repositories", presented at ICALEPCS 2021, Shanghai, China, Oct. 2021, paper MOPV034, this conference.

[10] `https://gitlab.com/tango-controls/rfc`

[11] `https://tango-controls.readthedocs.io/projects/rfc/en/latest/`.

[12] `https://datatracker.ietf.org/doc/html/rfc7764`

[13] `https://gitlab.com/gitlab-org/gitlab/-/blob/master/doc/user/markdown.md#front-matter`

[14] `https://datatracker.ietf.org/doc/html/rfc5234`

[15] `https://gitlab.com/tango-controls/rfc/-/blob/draft/README.md`

[16] `https://www.sphinx-doc.org`

[17] `https://myst-parser.readthedocs.io/en/latest/`.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL03

[18] https://tango-controls.readthedocs.io/projects/rfc/en/raw-rfc-10-request-reply/10/RequestReply.html

[19] https://gitlab.com/tango-controls/pogo

[20] https://gitlab.com/tango-controls/rfc/-/issues/128

[21] https://gitlab.com/tango-controls/rfc/-/merge_requests/58

[22] https://gitlab.com/tango-controls/rfc/-/issues?label_name%5B%5D=v10

[23] https://gitlab.com/tango-controls/rfc/-/merge_requests/39

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

CI-CD PRACTICES AT SKA

Di Carlo M.*, Dolci M., INAF Osservatorio Astronomico d'Abruzzo, Teramo, Italy
Harding P.[1], U. Yilmaz, SKA Organisation, Macclesfield, UK
Ribeiro B., Instituto de Telecomunicações Aveiro, Portugal
Morgado J. B., CICGE, Faculdade de Ciências da Universidade do Porto, Portugal

Abstract

The Square Kilometre Array (SKA) is an international effort to build two radio interferometers in South Africa and Australia forming one Observatory monitored and controlled from global headquarters (GHQ) based in the United Kingdom at Jodrell Bank. SKA is highly focused on adopting CI/CD practices for its software development. CI/CD stands for Continuous Integration & Delivery and/or Deployment. Continuous Integration is the practice of merging all developers' local copies into the mainline frequently. Continuous Delivery is the approach of developing software in short cycles ensuring it can be released anytime, and Continuous Deployment is the approach of delivering the software into operational use frequently and automatically. This paper analyses the decisions taken by the Systems Team (a specialized agile team devoted to developing and maintaining the tools that allow continuous practices) to promote the CI/CD practices with the TANGO-controls framework.

INTRODUCTION

When creating releases for end-users, every large software endeavour faces the problem of integrating different parts of their software solution and bring them to the production environment. When many parts of the project are developed independently for some time, an integration problem arises when merging them into the same branch, consuming more developer resources than originally planned. In a classic Waterfall Software Development process this is usual but also happens when following the classic Git Flow — also known as Feature-based Branching, which is when a branch is created for a feature. As an example, considering one hundred developers working in the same repository each of them creating one branch, merging can easily lead to conflicts becoming unmanageable, for a single developer to solve, thus introducing a delay in the releases (in literature this is called "merge hell"). This problem becomes evident especially working with over a hundred repositories with different underlying technologies. Therefore, it is essential to develop a standard set of tools and guidelines to systematically manage and control different phases of the software development life cycle throughout the organisation.

In the Square Kilometre Array (SKA) project, the selected development process is SAFe Agile (Scaled Agile framework) that is incremental and iterative with a specialized team (known as the Systems Team) devoted to supporting the Continuous Integration, Continuous Deployment, test automation and quality.

* matteo.dicarlo@inaf.it

Continuous Integration (CI)

CI refers to a set of practices requiring developers to integrate code into a shared repository often. Each commit is verified by an automated build, allowing teams to detect problems early in the process, giving feedback about the state of the integration. Martin Fowler [1] states various practices in this regard:

- maintain a single source repository for each system's component, favouring the use of a single branch;
- automate the build (possibly all in one command);
- automated test is run after build process for the software to be self-testing (this is crucial: all benefits of CI rely on the test suite being high quality);
- every commit should build on an integration machine: the more developers commit the better it is (common practice is at least once per day);
- frequent commits reduce potential conflicts: developer workflow is reconciled on short windows of change;
- main branch must always be stable;
- builds must be fast so that problems are found quickly;
- multi-stage deployment: every software build must be tested in different environments (testing, staging, etc);
- make it easy to get the latest version: all programmers should start the day by updating their local copies;
- Everyone can see what's happening: a testing environment with the latest software should be running.

Continuous Delivery & Deployment (CD)

Continuous Delivery [2] refers to a CI extension focusing on sustainably automating the delivery of new software releases. Release frequency can be decided according to business requirements, but the greatest benefit is reached by releasing as quickly as possible. Deployment has to be predictable and sustainable, irrespective of whether it is a large-scale distributed system, complex production environment, embedded system, or app. Therefore the code must always be in a deployable state. Testing becomes the most important activity, needing to cover enough of the codebase.

Often, the unsupported fact that frequent deployment equals lower levels of stability and reliability, is assumed. For software, the golden rule should be "if it hurts, do it more often, and bring the pain forward" — [2], page 26.

There are many patterns around continuous deployment related to the DevOps culture [3], "the outcome of applying

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

the most trusted principles from the domain of physical manufacturing and leadership to the IT value stream. [...] The result is world-class quality, reliability, stability, and security at an ever lower cost and effort; and accelerated flow and reliability throughout the technology value stream, including Product Management, Development, QA, IT Operations, and Infosec". It fosters increased collaboration between development (requirements analysis, development and testing) and operations (deployment, operations and maintenance) within IT. In the era of mainframe applications was common to have the two areas managed by different teams resulting in a development team with low interest in the operational aspects (managed by a different team) and vice versa. Sharing responsibility means that development teams share the problems of operations by working together in automating deployment operations and maintenance, and in return, operations have a deeper understanding of the applications being supported. It is also very important that teams are autonomous: being empowered to deploy a change to production with no fear of failure. This is only possible by supplying the necessary testing/staging platform and required infrastructure tools for developers to engage with the platforms and by using applications and deployment processes that can be rolled out and reverted if required.

Automation is one of the key elements of a DevOps strategy. It allows teams to focus on what is valuable (code development, test results, etc.) instead of the deployment itself, reducing human errors. The importance of those practises is to reduce risks of integration issues, releasing new software and creating better software products. CD goes one step further, as every single commit to the software that passes all the stages of the build and test pipeline, is deployed into the production environment — preferably automatically.

CONTAINERISATION

The *system engineering* development process was adopted in the initial design phase of the SKA project to reduce complexity by dividing the project into simpler elements. For every element, an initial architecture was developed, which comprises the software modules needed corresponding to a repository — each a component of the system.

Since all components need to be deployed and tested together, the first decision is how to package them. A container is a standard run-time unit of software that packages code and dependencies so that the component runs quickly and reliably across different computing environments. A *Docker* [4] *container image* is a lightweight, standalone, software package including everything needed to run an application: code, runtime, system tools, system libraries and settings.

One of the main dependencies in the SKA software is the TANGO-controls [5] framework, a middleware for connecting software processes mainly based on the CORBA standard (Common Object Request Broker Architecture). The standard defines how to expose the procedures of an object within a software process with the RPC protocol (Remote Procedure Call). TANGO extends the definition of an object with the concept of a Device that represents a real or virtual device to control. This exposes commands (procedures), and attributes (i.e. state) allowing both synchronous and asynchronous communication with events generated from attributes. Fig.1 shows a module view of the framework.

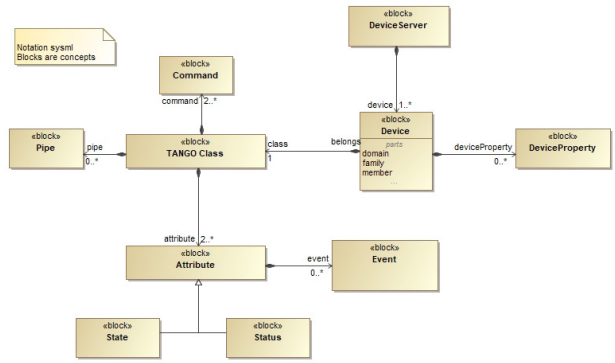


Figure 1: TANGO-controls simplified data model.

The importance of containers becomes clear with dependencies. The entire framework is packaged onto a set of containers [6] so that the final product is a containerized application that will be run in a system for managing these applications. Specifically, there is a SKA repository *ska-tango-images* [6], encapsulating all its components in a set of container images. Fig. 2 shows a simplified diagram for this project. By extending one of them, TANGO-controls becomes a layer inside the base images of any SKA module solving the dependency once for all.

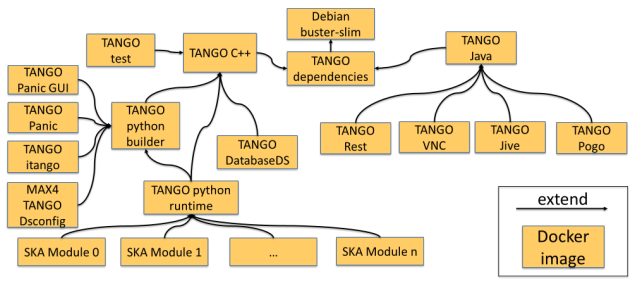


Figure 2: SKA-tango-images repository.

Kubernetes (K8s) [7] is used for container orchestration and *Helm Charts* [8] for declaring runtime dependencies for K8s applications. In K8s all deployment elements are resources abstracted away from the underlying infrastructure implementation. For example, a Service (network configuration), PersistentVolume (file-system type storage) or Pod (the smallest deployable unit of computing, consisting of containers). The resources reside in a cluster (a set of connected machines) and share a network, storage and other resources like computing power. Helm is a tool for managing K8s deployments with charts — a package of pre-configured K8s resources, tied to run-time instance configuration.

Namespaces create a logical separation of resources within a shared multi-tenant environment. A Namespace

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

enforces a separate network and set of access rights enabling a virtual private space for contained deployment.

ska-tango-images repository contains the definitions of two Helm Charts: *ska-tango-base* and *ska-tango-util*.

ska-tango-base: aplication Helm Chart defines the basic TANGO ecosystem with the following K8s services:

- *tangodb*: MySQL database used to store configuration data used at the startup of a device server;
- *databaseds*: device server providing configuration information to system's components, and runtime catalogue of the components/devices;
- *itango*: interactive TANGO client;
- *vnc*: Debian environment with x11 window system together with vnc [9] and noVNC [10] installed on it;
- *tangorest*: rest api [11] for TANGO eco-system;
- *tangotest*: TANGO test device server [12].

ska-tango-util: library chart helper for defining TANGO device servers on applications. Defines the following Helm templates:

- *multidevice-config*: K8s ConfigMap defining dsconfig JSON configuration file, bootstrap script for dsconfig, and python script for multi-class device server startup;
- *multidevice-job*: job for dsconfig application to apply a configuration JSON file set into the values file;
- *multidevice-sacc-role*: K8s service account, role and role binding that waits for configuration job to finish;
- *multidevice-svc*: K8s service and a K8s StatefulSet for a device server tag specified in the values file.

A Helm Chart contains at a minimum, information concerning the version of the container images and pull policy (image retrieval rule) for deployment. It also contains the necessary information to correctly initialize the TANGO database (device configuration) and how is exposed to other applications for discovery whiting the cluster.

Other SKA repositories *Makefiles* are selected as an abstraction and organisation layer eliminating language-specific scripts for building, testing, deployment and promoting ease of use in CI/CD. The use of a *Makefile* in each project simplifies containerisation work and automation of code building, testing and packaging processes, making it possible with a single command to compile the project, generate container images and test them by dynamically installing the related Helm Chart in a K8s environment.

The Makefile also enables publishing of container images and Helm Charts to SKA artefact repository and promotes reusability of the same build toolchain in different environments such as local development and CI/CD lifecycle.

ARCHITECTURE FOR INTEGRATION

In the previous section, was highlighted how the SKA project can be seen as a set of elements composed by a set of software modules corresponding to a repository. For each repository, one or more container images are built and pushed into the artefact repository (Nexus [13] box shown in fig. 5) while for each element, a Helm Chart is published into the same storage solution.

Since a Helm Chart can be in a dependency relationship with another chart, this concept can be used for integrating the various SKA elements which comprise the SKA MVP Product Integration (SKAMPI [14]) in a composable way representing the bulk of the effort for integrating all SKA's software sub-systems. Fig.3 shows a simplistic view of this above concept and its hierarchy.

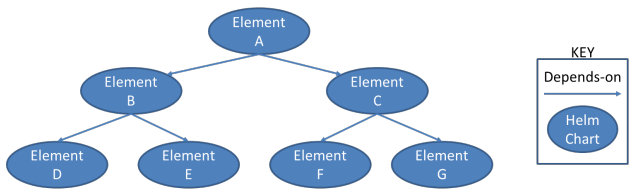


Figure 3: SKAMPI.

It is important to consider the operational aspects of the Helm dependencies which state that when Helm installs/updates a chart, the K8s resources from the chart and all its dependencies are aggregated into a single set, sorted by type followed by a name, and created/updated in that order. Due to this, a limit was imposed for single-level hierarchy with a parent chart, called the umbrella chart, which pulls together the charts of the hierarchy. While SKAMPI is the composition of the entire hierarchy, it is possible to think of different umbrella charts for other purposes like integration testing between a select few elements of the hierarchy. Fig. 4 shows the umbrella chart concept: the blue umbrella chart is the entire hierarchy while the red and green ones are for other purposes. This means that every SKA element can perform its integration testing by creating an umbrella chart with sub-elements needed for its integration.

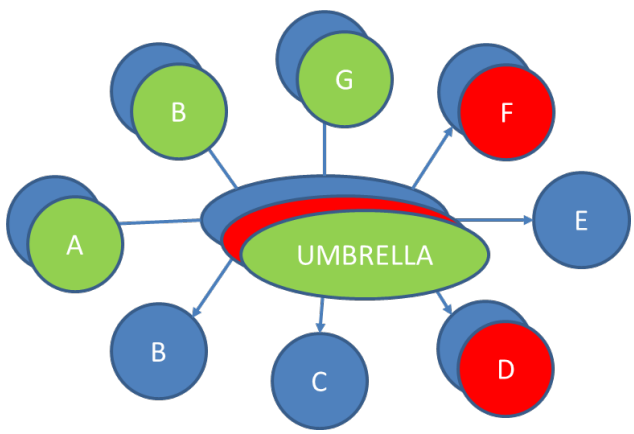


Figure 4: The umbrella chart concept.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

SKA INFRASTRUCTURE

To support the integration's architecture an infrastructure was built consisting of a standard footprint of VPN/SSH JumpHost gateway (called Terminus), Monitoring, Logging, Storage and K8s services to support the GitLab [15] runner architecture, and MVP testing facilities as shown in Fig.5 used to support DevOps and Integration testing facilities.

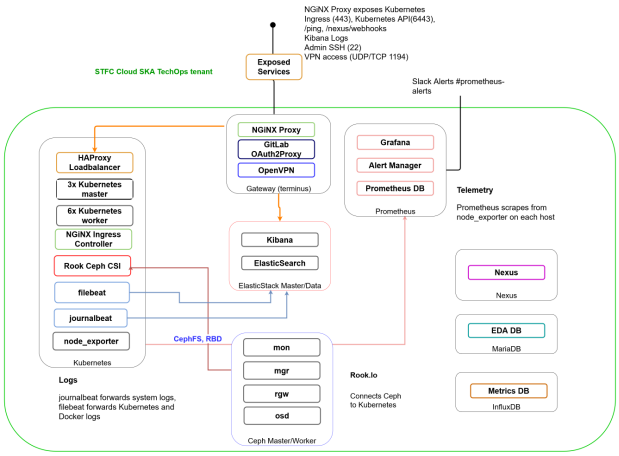


Figure 5: STFC cloud components.

A K8s cluster was deployed with 1 LoadBalancer, 3 Master, and 6 Worker configuration. Fig.6 illustrates how the LoadBalancer ties K8s services together exposing deployed applications to the outside world. The K8s API Server is exposed externally from Terminus using TCP pass-through, and NGiNX [16]'s Ingress Controller is SSL terminated for external access. These services are exposed using an NGiNX reverse proxy. Ingress access on port 443 is password protected using oauth2-proxy [17] integration with GitLab.

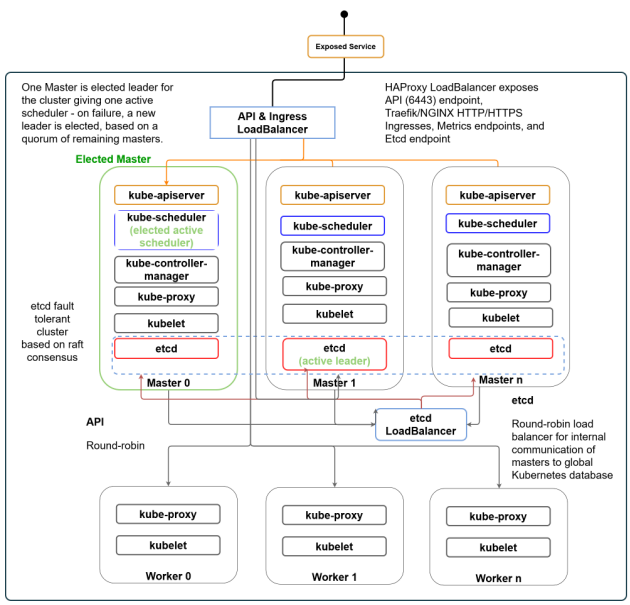


Figure 6: Kubernetes components.

In Fig.7 it is hown the K8s runner [18] works as a multiplexer receiving requests from GitLab for jobs and launching their respective Pods up to a configured scaling limit (15 cur-rently). GitLab's runners use intermediate cache to speed up jobs by passing dependencies between them. This cache is based on Minio [19] with S3 [20] compatible buckets for storage.

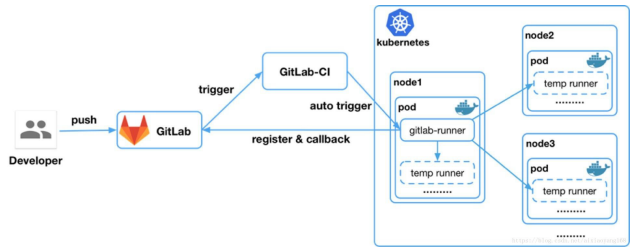


Figure 7: GitLab runner.

PIPELINE

To bring everything together for a complete CI/CD toolchain, GitLab [15] was selected. The data model for a generic SKA software is shown in Figure 8.

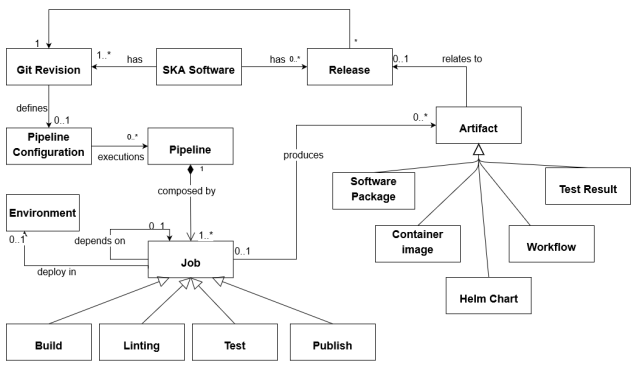


Figure 8: Pipeline definition data model.

The entry point of the diagram is the Pipeline box, composed of several jobs. This was standardised for each project regardless of its artefacts so that the same standardised steps for code/configuration and Helm Charts are followed:

- **linting**: code is analysed against sets of coding rules to check if it follows the agreed best practices;
- **build**: code is compiled and a container image created;
- **test:** compiled package (and image) are tested;
- **publish**: code artefacts are published;
- **pages**: test results are published (GitLab's naming).

The pipeline is respected as the main hub of software development in which code is built, tested, verified, published and integrated. These steps are used in local development (where the same shell scripts are available thanks to the *Makefile* targets), merge workflow, QA, integration and release. Also, having an almost identical platform environment

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

for different stages of the software lifecycle, significant differences between development and operations are eliminated.

Fig.9 shows a simplification of the run-time behaviour of the selected technologies working together. At the centre part, there is a K8s cluster defined for every project in SKA's telescope. Outside the K8s cluster, are GitLab code repositories and pages, Nexus Artifact repository [13] (store packaged code artefacts), ELK stack (Elasticsearch, Logstash and Kibana) [21] for logging, prometheus [22] for monitoring (metric collection) and Ceph [23] for distributed storage.

Inside the cluster, in an isolated K8s Namespace, are the GitLab runners related K8s resources, checking every 30 seconds if there are pending pipelines triggered manually or pulled by resources. If the runner finds a pipeline, it creates a K8s Pod for each job defined in the configuration file. Each created Pod is capable of deploying an umbrella chart needed for specific testing of the repository in an isolated Namespace. The installed deployment can then be tested and the job's result will be reported to GitLab producing artefacts, to be stored in the correct artefact repository.

During any stage of the pipeline, jobs can download required dependencies from the artefact repository. Depending on the type of job, the pipeline is used for deploying the permanently running version of SKAMPI or other resources needed. The K8s cluster is equipped with monitoring solutions to examine the cluster's health and performance and any resources that are deployed in it. Storage and logging solutions are integrated to provide a consistent logging and distributed storage framework for the resources. Finally, this architecture for creating temporal K8s resources for pipeline steps (testing, building, packaging, etc.) ensures that necessary environments for the jobs are always clean — i.e. not affected by previously run pipelines.

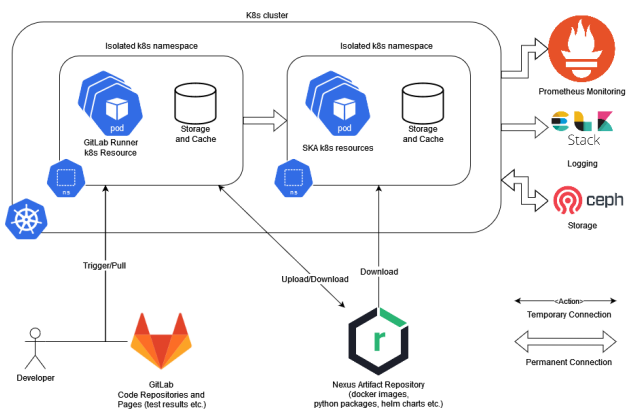


Figure 9: CICD at run time.

TESTING

The most important in CI is testing, so we need to question ourselves how a generic component of the SKA can be tested with the above architecture. At the SKA, testing was split into two distinct types: pre-deployment and post-deployment. Deployment happens when a runner executes a job with a GitLab environment keyword. By doing so, the job is linked to the K8s cluster through GitLab configuration. While pre-deployment tests (unit tests) are made without the real system online (using stubs and mocks), other tests (integration and system tests) need more than one live system component to be up and running as they will be using other services and applications. The SKA is composed of several distinct modules, each of them with its repository and different requirements for the components needed for integration and system testing. For each, an umbrella chart was introduced which enabled the specific component to be deployed together with its dependencies. Specifically, to enable the GitLab pipeline to deploy and test the chosen component each repository must:

- contains at least one Helm Chart;
- has an environment;
- has a Makefile for K8s testing.

A set of templates and standardized Makefiles was developed by the System Team, so it is only necessary to include them in the repository. The post-deployment test job is then composed of the following steps defined in those Makefiles:

- **install**: installs chart (and sub-charts) in the Namespace specified in the environment;
- **wait**: wait for every container to be running;
- **test**:
  - create container in the Namespace specified in the environment;
  - run PyTests inside this container;
  - return test results.
- **post test**: delete all resources allocated for tests.

The artefacts are the output of the tests, containing reports both in XML and JSON and other (i.e. PyTest's) output so that consequent pipeline steps (mostly packaging and releasing) can be run.

DEVELOPMENT WORKFLOW

There are two important assumptions for understanding SKA's development workflow: the master branch shall always be stable, and branches shall be short-lived. Stable means that the master branch always compiles, and all automated tests run successfully. This also means that every time a master branch results in a condition of instability, reverting to a condition of stability shall have precedence over any other activity on the repository. As a result, the selected development workflow for SKA is the following:

- developer works on current code base's copy;
- work on a story starts on a new branch named after it;
- developer frequently commits changes to local git repo;

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

- developer creates unit tests to be run on local environment until they successfully pass;
- once tests pass, changes are pushed to a remote branch;
- CI server (GitLab):
  - checks out changes when they occur;
  - runs static code analysis providing feedback;
  - builds system and runs unit and integration tests on the branch;
  - provide feedback to developer about test status;
  - provide feedback about coverage metrics.
- once all branch's tests execute successfully, the developer opens a pull request (*i.e. GitLab's Merge Request*) for merging changes onto master;
- code is reviewed and approved by other developers;
- code is merged into the master branch;
- CI server (GitLab):
  - runs whole pipeline again including the tests on master branch;
  - releases deployable artefacts for testing (reports, code analysis, etc.);
  - assigns build label the code's version built (i.e. docker image version);
  - alerts team if build or tests fail for them to fix issue ASAP;
  - publishes the successful build artefacts to artefact repository.

CI-CD AUTOMATION AND QUALITY

To verify if all best practices are followed, plugins and tasks were implemented to perform quality checks on GitLab's merge requests and artefacts published to Nexus. Fig.10 shows a module's view of the frameworks for those.

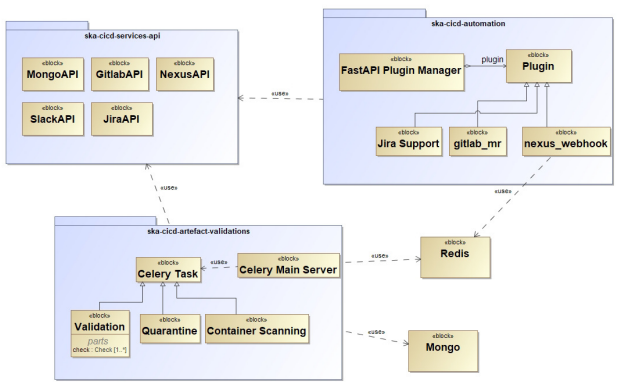


Figure 10: CI-CD automation framework.

The above diagram shows three main packages:

- **ska-cicd-services-api** [24] which includes all APIs that will be used in the other two packages. These APIs allow to communicate with GitLab (i.e merge request creation or for project information), Slack [25] (i.e. sending messages to channels), Jira [26] (i.e. project information) and Nexus (i.e. obtaining component information);
- **ska-cicd-automation** [27] which uses FastAPI [28] to build a Python web application with a plugin architecture. Three plugins have been created: **gitlab_mr** used for merge request quality checks and providing feedback to developers directly on GitLab; **jira_support** to handle jira operations and **nexus_webhook** to trigger webhooks every time a new artefact is published;
- **ska-cicd-artefact-validations** [29] based on Celery [30] containing a main server pulling messages from Redis [31], transforming them into artifact validation tasks and storing the results into a MongoDB [32] database.

Merge Request Quality Checks

To ensure that every developer follows the development workflow and best practices, automated checks are performed on Merge Requests. A webhook was added to the ska-telescope GitLab group, which triggers a service (fig. 10 *ska-cicd-automation* [27]) every time a new Merge Request is created. The quality checks will then verify if:

- the Merge Request Settings were set correctly;
- the branch name, the commit messages and the merge request have a Jira Ticket ID;
- the project has a proper license;
- the project as documentation on it and if it was updated;
- the project has pipelines with the needed jobs.

After performing the checks, their result is reported back to the developers on GitLab's main Merge Request page via a comment (see Fig. 11 for an example of the comment, includ-ing a table with the severity of the failed check, description about the check and mitigation strategy).

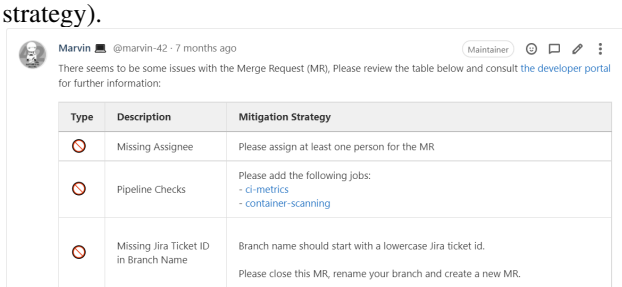


Figure 11: Checks results table.

Nexus Artefact Validation

There are many packaged code artefacts of multiple formats being created in the SKA project hosted on GitLab

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

and then published to the Nexus Artifact repository. Those should follow SKAO's conventions: Artefact names should be compliant with semantic versioning 2.0.0 [33] and must include associated metadata with the required information, such as who published it, from which GitLab repository is they originate from and other useful information.

To ensure that the guidelines and policies described are followed for consistent, compliant and robust artefact management, there are a series of automated validations in place at *ska-cicd-artefact-validations* [29]. If an artefact fails validation, it is moved to a quarantine state and the results of the checks are reported back to developers that triggered the pipeline that published it. This report is made by creating a new Merge Request where the developer is made assignee and its description contains a table composed of the failed validations and instructions on how to mitigate them.

The execution of artefact validations happens following the Celery architecture with a server that pulls messages from a queue (Redis) and creates tasks (processes) to perform the specific validation. Every task can then create other tasks as needed to perform other activities (i.e. quarantine the artefact or create merge request on GitLab). The result validation is stored in a MongoDB database.

When an artefact is published on the GitLab job the following tasks are performed:

- **validation**: performs the validation checks;
- **get metadata**: extract existing metadata from artefacts;
- **container scanning**: scans for container vulnerabilities using Trivy;
- **quarantine**: quarantine artefacts if any checks fail;
- **create MR**: create MR to report failures to developers;
- **insert DB**: insert metadata into MongoDB about the validation performed.

With the above tasks, it is possible to keep and maintain a clean and organized repository, where all artefacts follow the guidelines and policies defined on the project.

CONCLUSION

The majority of decisions taken by the Systems Team follow the workflow described by the Continuous Integration process outlined in Martin Fowler's paper inspired by the state-of-the-art industry practices of [1–3]. In particular:

- for each component of the system, there is only one short-lived repository with minimal use of branching;
- artefact build, tests and publishing are automated with the use of few commands (Makefile targets);
- every commit triggers a build in a different machine;
- once artefacts are built, the SKAMPI will automatically create a new deployment of the system and more tests will be done at that level (i.e. system tests);
- having a common repository (Nexus and GitLab pages) for the code artefacts and test results simplifies downloading the latest changes from every team and for each component to enable fast development;
- the integration environment is accessible to all developers and deployed in a unique Namespace on the cluster.

In addition, every artefact is validated in terms of quality so that a common standard across the project is maintained.

ACKNOWLEDGEMENTS

This work was supported by Italian Government (MEF - Ministero dell'Economia e delle Finanze, MIUR - Ministero dell'Istruzione, dell'Università e della Ricerca).

REFERENCES

[1] Martin Fowler, Continuous Integration, https://martinfowler.com/articles/continuousIntegration.html

[2] J. Humble, D. Farley, "Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation", 2010, ISBN (0321601912, 9780321601919), Pub. Addison-Wesley Professional

[3] G. Kim, P. Debois, J. Willis, J. Humble, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations", ISBN (1942788002 9781942788003)

[4] Docker, https://www.docker.com/

[5] TANGO-controls, https://www.tango-controls.org/

[6] ska-tango-images repository, https://gitlab.com/ska-telescope/ska-tango-images

[7] Kubernetes, https://kubernetes.io/

[8] Helm, https://helm.sh

[9] v11vnc, https://github.com/LibVNC/x11vnc

[10] noVNC, https://github.com/novnc/noVNC

[11] TANGO-controls REST API, https://gitlab.com/tango-controls/rest-api

[12] TANGO-controls test device server, https://gitlab.com/tango-controls/TangoTest

[13] Nexus, https://www.sonatype.com/nexus/repository-pro/

[14] SKAMPI - SKA Mvp Prototype Integration, https://gitlab.com/ska-telescope/ska-skampi

[15] GitLab, https://gitlab.com/

[16] NGINX, https://www.nginx.com/

[17] OAuth2 Proxy, https://github.com/oauth2-proxy/oauth2-proxy

[18] GitLab Runner, https://gitlab.com/ska-telescope/sdi/deploy-gitlab-runners/

[19] MinIO, https://operator.min.io/

[20] Amazon S3 buckets, https://aws.amazon.com/it/s3/

[21] Elasticsearch, https://www.elastic.co/

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL04

[22] Prometheus, https://prometheus.io/

[23] Ceph Storage, https://ceph.io/

[24] ska-cicd-services-api, https://gitlab.com/ska-telescope/sdi/ska-cicd-services-api

[25] Slack, https://slack.com

[26] Jira, https://www.atlassian.com/software/jira

[27] ska-cicd-automation, https://gitlab.com/ska-telescope/sdi/ska-cicd-automation

[28] FastAPI, https://fastapi.tiangolo.com/

[29] ska-cicd-artefact-validations, https://gitlab.com/ska-telescope/sdi/ska-cicd-artefact-validations

[30] Celery, https://docs.celeryproject.org/en/stable

[31] , Redis, https://redis.io/

[32] MongoDB, https://www.mongodb.com

[33] Semantic versioning 2.0.0 https://semver.org/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

Pysmlib: A PYTHON FINITE STATE MACHINE LIBRARY FOR EPICS

D. Marcato[1,3,*], G. Arena[1], M. Bellato[2], D. Bortolato[1], F. Gelain[1], G. Lilli[1], V. Martinelli[1], E. Munaron[1], M. Roetta[1], G. Savarese[1],
[1]INFN Legnaro National Laboratories, 35020 Legnaro, Italy
[2]INFN Padova Division, 35131 Padova, Italy
[3]Department of Information Engineering, University of Padova, 35131 Padua, Italy

Abstract

In the field of Experimental Physics and Industrial Control Systems (EPICS) [1], the traditional tool to implement high level procedures is the Sequencer [1]. While this is a mature, fast, and well-proven software, it comes with some drawbacks. For example, it's based on a custom C-like programming language which may be unfamiliar to new users and it often results in complex, hard to read code. This paper presents pysmlib, a free and open source Python library developed as a simpler alternative to the EPICS Sequencer. The library exposes a simple interface to develop event-driven Finite State Machines (FSM), where the inputs are connected to Channel Access Process Variables (PV) thanks to the PyEpics [2] integration. Other features include parallel FSM with multi-threading support and input sharing, timers, and an integrated watchdog logic. The library offers a lower barrier to enter and greater extensibility thanks to the large ecosystem of scientific and engineering python libraries, making it a perfect fit for modern control system requirements. Pysmlib has been deployed in multiple projects at INFN Legnaro National Laboratories (LNL), proving its robustness and flexibility.

INTRODUCTION

The Experimental Physics and Industrial Control Systems (EPICS) [1] is one of the most successful frameworks to develop control systems for physics facilities, being used at major laboratories and experiments all around the world. Its main feature is the implementation of the Channel Access (CA) (and the PV Access in newer versions), a standard protocol where the different parts of the control system can communicate. This provides a standard interface to access all the Process Variables (PV) and works as a hardware abstraction layer. Using this protocol, many components have been developed by the community to provide the core functionalities of a modern control system, like Phoebus [3] and React Automation Studio [4] for Graphical User Interfaces (GUI) or the Archiver Appliance [5] for historical data storage.

The sequencer is the tool proposed by the EPICS core developers to implement high level procedures and Finite State Machines (FSM) for process automation. This is an extension of the core software and was first proposed in 1991 in the EPICS paper [1] and originally developed at the Los Alamos National Laboratory. It defines a C-like language called State Notation Language to develop finite state machines which is transcompiled to C code and then compiled to machine code. The user can define states and transitions, while the sequencer takes care of low-level details like the connection with the Channel Access, the event handling and concurrency. Finally, the code is usually run as part of a EPICS input output controller (IOC), which is the piece of software which defines and publishes the PVs on the Channel Access.

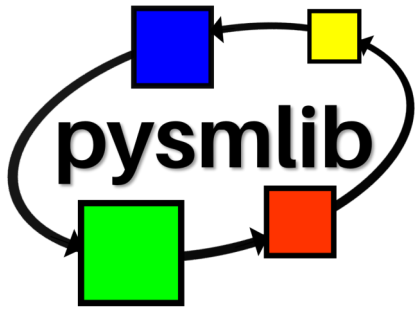


Figure 1: Pysmlib logo.

This software has proved valuable and has been widely adopted thanks to its good performance, seamless integration with the Channel Access and the IOC, and its programming model. Even so, some of its limitations have emerged over time. For example, being C-based was a great advantage at the beginning since it means that one could extend it with any C/C++ library. Today, higher level languages are preferred for this kind of high level tasks, and performance is no longer a limiting factor in most cases. For this reason Python has emerged as one of the most prominent languages for modern scientific and engineering computing, thanks to a large number of dedicated libraries. Also, Python appeals to a broader audience of less-technical programmers.

The PyEpics [2] python library, which wraps the original `libca` C library, became thus a popular alternative to communicate with the Channel Access. This can be used to write both simple scripts and full blown programs. Large experiments or collaborations used this, or similar wrappers, to build tightly integrated high level suites which handle automation and much more at facility level, such as ophyd and bluesky [6]. These are great solutions, but require a big investment into their design model, which could not be ideal for simple tasks or small independent laboratories. Also, at this level there is a lot of fragmentation in the community, with no default go-to solution but many different approaches tailored to the needs of specific laboratories.

* davide.marcato@lnl.infn.it, www.davide.marcato.dev

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

In the middle between vanilla PyEpics scripts and full experiment-wide software, there is still the need for a generic standalone solution to develop finite state machines, where the execution flow, event handling and connection with the channel access is provided. Pysmlib aims to recreate the sequencer programming model and its strengths, while taking advantage of the python language and focusing on ease of use.

This paper formally presents the library to the scientific community, describing its design and going into details on the implementation and execution flow. Then an overview of the user interface is presented and finally the current status of the project and some plans for the future are outlined.

DESIGN PRINCIPLES

A Finite State Machine is a computational abstraction where the program behaviour depends on its current state and the input it receives. The machine can perform a transition to a different state in response to inputs. For example the schema in Fig. 2 represents a simple FSM with 3 states S_1, S_2, S_3 and 2 inputs x_0, x_1. The execution starts from S_1 and while x_0 is 0 the current state does not change. When x_0 becomes 1, the FSM executes a transition to S_1. Now the FSM follows the value of x_1 by staying in S_1 when its value is 0 and moving to S_2 when it becomes 1. In S_2 the FSM can go back to S_0 if x_0 becomes 0 or to S_1 when x_1 is 0.

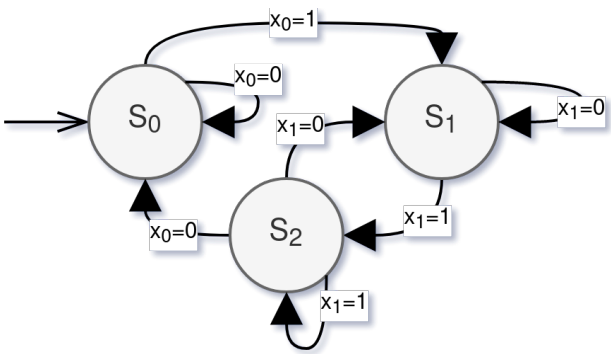


Figure 2: Schematic representation of a FSM with 3 states and 2 inputs.

This kind of abstraction is extremely useful to design control system algorithms, where the controlled system can be described with states and transitions, and the control algorithm can be tuned for the current state.

Pysmlib aims to provide a simple way to describe Finite State Machine states and transitions, while handling all common tasks and minimizing the custom code required. Thus, it keeps track of the current state and executes it in response to events and takes care of connecting the FSM to the inputs, while providing the user some methods to access them. The term *input* is here used both for input (readable) and output (writable) signals. In fact, in pysmlib they are an abstraction of Channel Access PVs, even though the library is designed so that a future expansion to different types of inputs is relatively easy and requires as little updates to the user interface as possible. On top of the methods to read and write PVs provided by PyEpics, the methods to detect rising and falling edges or changes are provided, which are particularly useful for the design of a FSM. Then, the user can implement the FSM states as methods of a class, which represents the FSM, with a specific naming convention and the library is able to automatically discover them. When performing a transition from one state to another, the user can decide to implement *entry* and *exit* methods for each state, which are executed only once during the transitions, as can be seen in Fig. 5.

Pysmlib is designed to maximize network efficiency and system responsiveness, in order to be able to react with minimum latency to input events. Thus, all the inputs are connected at startup time, and the connection is kept in memory during the whole execution. The current state is executed every time a channel access event is received, in a event driven way, without relying on a periodic loop. For example, when an input changes its value or its connection status, the current state is re-evaluated so that the FSM can react to the new value, while the state is never executed if no event is received. To optimize network usage, multiple FSMs can be loaded and run in multi-threading while sharing the connection to common PVs and a dispatcher is used to broadcast channel access events to each FSM. The user is expected to develop multiple FSMs, load them together on a single *daemon*, using the provided loader helper, and let it run continuously. This approach comes natural for always-on operations (eg: a feedback correction) but it is also the ideal execution flow for procedures that start from a user input or some external conditions: in these cases the FSM will wait on a *idle* state where no action is performed until the enable signal is received, execute its core procedure, and then come back to idle. This means that when the enable arrives, all the inputs are already connected and the user doesn't have to wait for the FSM startup and all the connection times.

To ensure consistency, the inputs should not change value or status during the execution of a state. For this reason a queue of events is used and they are evaluated in order one by one, where each event triggers one execution of the current state of all the FSMs connected to such input. Along with input events, the user can also set *timer* events, so that the current state is executed after a custom amount of time. This is useful to set timeouts or to execute periodic actions. Other features include logging methods to write FSM logs in a unified way, and a watchdog implementation. In fact, contrary to the sequencer where the code is usually executed on the same IOC that defines the PVs, here the FSM daemons can run everywhere on the channel access network, and most probably will run independently from the IOC and from the user interface. A method to assert if the FSM is online and running is thus needed, since the control system may rely on the FSM to work correctly. For this reason a watchdog is implemented, where the FSM will periodically write to an external PV. The PV is then configured to raise an alarm or an error when the write operation does not happen regularly.

Finally, pysmlib is developed as a generic library with no dependency on the task to perform, and aims to be useful

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

for a large audience of control system developers. It offers an online documentation and follows open source standards, where the users are encouraged to contribute.

SOFTWARE ARCHITECTURE

In this section the internal architecture of the library is detailed, along with a description of the execution flow of a FSM. Figure 3 shows a summary of the main classes with their attributes and methods. These can be grouped into 4 modules, as highlighted by the color of the tables. Each module concerns with one of the main features of the library: access to the inputs (green), FSM execution (blue), timers (orange) and other smaller utilities (red). A complete description of each module follows.

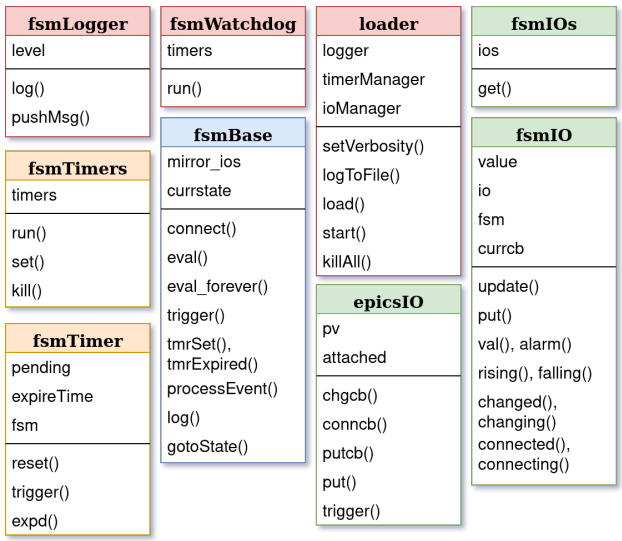


Figure 3: Diagram of the main classes and their most important attributes and methods.

Input Management

The Channel Access defines some methods to access the Process Variables: it's possible to perform a `get` (read) or a `put` (write) operation, but also to `monitor` a PV, that is register to the IOC server to be notified about PV events. There are 3 types of events: *change* events are emitted when the PV has a new value, *connection* events when the connection status changes and *put complete* events when a put operation completes. With PyEpics it's possible to register a callback to be executed when one of these events occurs, so that the event can be processed.

Pysmlib defines the `epicsIO` class which is responsible to keep the connection with a PyEpics `PV` object and registers the callbacks on the Channel Access events. When an event arrives a `trigger` method is executed, which will pass a copy of the event data to all the attached FSMs. Each FSM has a queue of events where this data is temporarily placed, while waiting to process it. All of this is done on the callback thread, while the queue consumption is done on the FSM execution thread, so the queues must be thread safe. Furthermore, this class is also responsible to perform the `put` operations, which are directly executed by the FSM, since the Channel Access already implements a queuing system. The `epicsIO` class is instantiated when a FSM connects to a new PV. To avoid duplicating connections to the CA when different FSMs connect to the same PV, a helper class (`fsmIOs`) is used. This class keeps track of the available inputs and creates a new `epicsIO` instance only when required.

One important requirement is that an input does not change during the execution of a FSM state. To achieve this, every time a FSM connects to an input it creates a corresponding instance of `fsmIO`, a class which represents a local copy of an input to each FSM. The execution follows this pattern: first an event is removed from the queue, then the local copy of the corresponding input is updated using the event data and finally the current state is executed. This pattern is repeated until there are no more events in the queue and the thread waits idle for new ones. So each event brings the update of a single input and triggers a single execution of the current state. If the original IO changes during the state execution, the local copy is not affected.

Figure 4 shows an example of the FSM execution flow in response to a *change* event from an input. First, the registered callback is executed by the PyEpics library and all the FSMs connected to the corresponding input are triggered. Then, the event data is queued on the event queue and the thread of the callback (blue in the figure) completes its execution. Now, the FSM thread (green in the figure) awakes and consumes the available events one by one. In order to keep the instance of `fsmIO` synchronized with the original input, first the `reset()` method is called, which removes old data, and then the `update()` one which feed the new information. Finally, the current FSM state is executed by calling the user-defined methods.

In the code, the user can inspect the new received value thanks to the `fsmIO` class, which implements all the methods to read and write the value and metadata of the input. For example, it's possible to read the current value with `val()`,

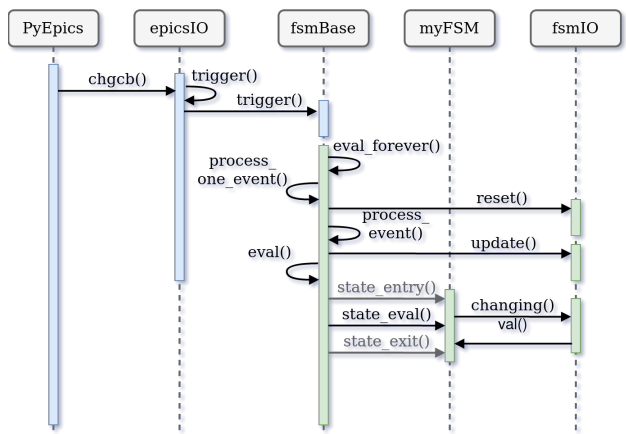


Figure 4: Execution flow of the FSM current state in response to an event from an input.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

connection status with `connected()` and the alarm severity with `alarm()`, and to write the value with `put()`. Other methods allow to detect rising or falling edges, or other transient properties of the input. For example, we want that the `connecting()` method returns `True` only during the state execution that was triggered by the connection event, and not on the following ones. Conversely, the `connected()` method will continue to return `True` until the input disconnects. This is especially useful for the design of a FSM to perform some actions just once when a condition is met. To implement this, the type of the current event is used to understand what kind of event triggered the state execution. For example, an input is `changing()` when the current event is of *change* type, while it is `rising()` when the current event is of *change* type and the new value is greater than the preceding one.

Finite State Machine Execution

The FSM execution flow is managed by the `fsmBase` class. The user is expected to derive from this class to implement his own specific FSM. In the constructor of the derived class the user will connect to all the inputs using the `connect()` method, and will specify the first state to execute. This can be done using the `gotoState()` method which accepts a string with the state name as argument. After that the user will implement the states as methods of the class. As illustrated in Fig. 5, for each state the user will implement a *eval* method and optionally the *entry* and *exit* methods which are executed only once during the state transitions. To be correctly recognized, the methods must be called by concatenating the state name to `_eval()`, `_entry()` or `_exit()`. For example, for a state called *idle*, the user must define the method `idle_eval()` and can optionally define the methods `idle_entry()` and `idle_exit()`.

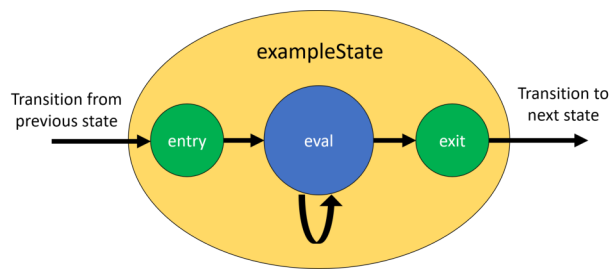


Figure 5: Entry, eval and exit methods can be implemented for each FSM state.

The `fsmBase` defines a thread which runs the FSM states. This thread initially waits on the event queue for some events to populate it. When an event arrives, it is removed from the queue and processed in order to update the `fsmIO` instance of the corresponding input. Then the `eval()` method is called, which is responsible for the actual state execution. This method performs the following steps:

1. Perform a state transition if required. In this case it also executes the `_entry()` method of the new state, if it's defined.
2. Execute the `_eval()` method of the current state.
3. If the user requested a state transition, the `_exit()` method of the current state is executed. In this case go back to step 1 without processing a new event.

The user can call `gotoState(new_state_name)` inside the state `_eval()` method to perform a transition. In this case, the execution loop is restarted from step 1 without processing a new event, so that the new state is evaluated once without waiting. The `gotoState()` function simply saves the name of the next state and uses the `getattr` python function to find the methods with the corresponding names. The actual transition is executed in step 1 by updating the pointer to the current state.

Other methods of the `fsmBase` class can be used to check conditions on multiple inputs together or to retrieve the input responsible for the current execution of the state. Furthermore, the class is used as a single interface for timers and the logger, as explained in the following paragraphs.

Timers

Timers are used to schedule internal events, where the current state is executed after a fixed amount of time. They can be used to perform asynchronous operations such as periodic writes or to enforce a timeout when waiting for an external condition to become true. The user can set a timer with the `tmrSet()` method of `fsmBase`, selecting an identifying name and an expiration time in seconds, while the methods `tmrExpired()` and `tmrExpiring()` can be used to verify if a timer has expired in the past or in the current event.

A single timer is represented by the class `fsmTimer`, which keeps track of the expiration status. The class `fsmTimers`, instead, is used to manage all the timers of all the FSMs. This is a thread which keeps a list of timers ordered by their expire time. When the FSM sets a timer, this is added to the list in the correct place. The thread schedules a waiting time equivalent to the remaining time of the first timer in the list, and then triggers the corresponding FSM by appending a *timer expired* event to its event queue. After that it removes the timer from the list and schedules a new waiting time. When the list is empty the thread waits idle until a new timer is set.

All the timers, inputs, and each FSM live on different threads, and thus create a complex multi-threading execution flow. With `pysmlib` this complexity is completely hidden from the user, which gets a well-tested core to build upon, without worrying about low level details.

Utilities

On top of the core functionalities, the library offers useful utilities to simplify the user code on some common tasks. For example, the `loader` class provides methods to create a launcher file which executes many FSMs in parallel. In fact, after the development of the FSM classes, it's often useful to run different instances of the same FSM with different

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

parameters, or simply to launch together correlated FSMs to share common resources. The `load` method of the `loader` class takes care of instantiating a FSM. It lets the user specify custom parameters to be passed to the constructor of the FSM, but it adds some system parameters used to share common resources. For example, a single instance of `fsmIOs` is created and passed to all loaded FSM in order to share the inputs. The same mechanism is used to share the timer manager thread.

The `loader` class is also used to configure log messages. In fact, the library provides an abstraction of a generic log facility which can be configured to send messages to different backends with a unified interface. At load time the user can specify to log to file (`logToFile()`) or standard output (default) and can set the verbosity level with the `setVerbosity()` method, choosing between *debug, info, warning* and *error* levels. Likewise, the `fsmBase` class provides the `logD()`, `logI()`, `logW()`, `logE()` methods that can be used to write log messages with the corresponding level. The logging facility is provided by the `fsmLogger` class, which is currently a custom class with few options, but can easily be expanded or replaced by the `logging` module of python without any changes to the user interface, thus minimizing the impact on the existing code.

Another shared resource is the `fsmWatchdog` class, which provides the ability to periodically write to an external PV while a FSM is running. This class is derived from `fsmBase` and implements a specific FSM which periodically writes the watchdog of all running, user-defined FSMs. The user can define an input of an FSM to be used as the watchdog PV by calling the `setWatchdog()` method of `fsmBase` on the constructor of each FSM, and this will be automatically passed to an instance of `fsmWatchdog` at load time. The user can also specify the interval and the value to be written for each watchdog, choosing between `0`, `1` or an intermittent value. The external PV should be configured so that when a write operation does not occur after a specified time interval, it raises an alarm. This can easily be achieved with the `HIGH` field of a binary output PV.

DEVELOPMENT TOOLS

The development of the library adheres to modern standards for python projects. The code is well formatted and documented, respecting the `pylint` indications. To ensure easier updates to the core library automated tests have been developed. These simulate the most common operations and check if the expected result is achieved, so that the developer can easily notice when an update or a new functionality introduces bugs. The tests were developed using the `pytest` [7] framework, and can be run directly with pytest or using `nox` [8], a tool to automate test executions in different environments. In fact, given a list of python versions, it takes care of creating a different conda environment for each version, install the dependencies and run the tests. This way, it's easy to verify that the code continues to run correctly on all supported versions of python. To simulate Channel Access connectivity the `pcaspy` module is used, which helps running a PV server from python, without firing up a complete EPICS IOC. Finally, the nox tests are run automatically on the Gitlab CI [9] every time the code is pushed to the server, so that the developer can be alerted of failures.

Since the library was developed with a broad and generic audience in mind, it has also been published as free and open source code on Github [10] with a GPLv3 license, which is compatible with the EPICS open license. The library is packaged as a standard python module using `setuptools` and is available on the PyPI [11] python software repository, so that it can be installed with just one command. Version strings are managed using `versioneer` [12], a tool that helps defining automatically the current version based on the underlying git tag.

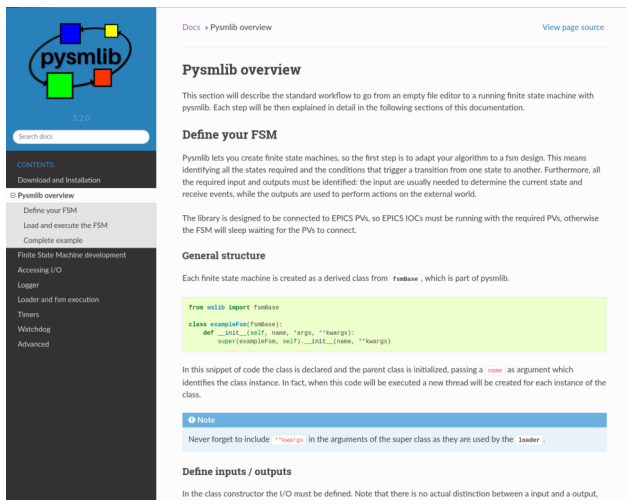


Figure 6: Pysmlib online documentation.

Web-based documentation was built using `sphinx`, which generates beautiful HTML pages from reStructuredText files. The resulting website can be seen in Fig. 6 and includes some examples, a tutorial, and the detailed API description. Finally, the HTML pages are hosted using Github Pages [13], so that they are always available online.

USER EXPERIENCE

Pysmlib was first envisioned and developed for the radio frequency (RF) control system [14] of the ALPI linear accelerator at the INFN Legnaro National Laboratories (LNL). This system requires a lot of procedures to power up, tune and lock each cavity which were previously performed manually. With this library it was possible to automate lots of tasks, thus reducing the time and effort required to setup the accelerator for a run.

Given the success in its first use case, the original code was decoupled from the RF control system to create a general purpose library that could be easily used on different projects. Thus, version `v2.0` introduced the *pysmlib* name and the library was published online, complete with documentation. Version 3 dropped the support for Python 2, which was being deprecated, and further consolidated the user interface,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

adding methods to access the alarm state of an input and improving the automated tests.

The library has been integrated into multiple projects at Legnaro to automate different tasks, from simple ones like acquiring the spectre of the beam using a magnetic dipole to more sophisticate ones like online beam emittance measurement or BOLINA [15] beam trajectory optimization algorithm. Another common use case is to build simulators, where the FSM reacts to control PVs simulating the effect of a real object, which is useful to test the control system software. For example, a FSM was developed to simulate a wire scanner diagnostic complete with its handling motor. A FSM can also be used to build alarm handlers, like `telegram-epics-bot` [16] which sends a Telegram message to selected users when a PV enters the alarm state. In general pysmlib is useful whenever it is necessary to develop programs which are not executed one-shot, but are always running waiting for the correct condition to execute their task. In fact, the ability to detect edges on the PV values is extremely useful for this scenario and with this library it is provided by default.

In listing 1 an example of a minimal complete FSM is provided, where the FSM is expected to copy the value of a *counter* PV to a *mirror* one, when the *enable* PV is non-zero.

```
#! /usr/bin/python
from smlib import fsmBase, loader

# FSM definition
class exampleFsm(fsmBase):
  def __init__(self, name, *args, **kwargs):
    super().__init__(name, **kwargs)
    self.counter = self.connect("counter_pv_name")
    self.mirror = self.connect("mirror_pv_name")
    self.enable = self.connect("enable_pv_name")
    self.gotoState('idle')

  # idle state
  def idle_eval(self):
    if self.enable.rising():
      self.gotoState("mirroring")

  # mirroring state
  def mirroring_eval(self):
    if self.enable.falling():
      self.gotoState("idle")
    elif self.counter.changing():
      readValue = self.counter.val()
      self.mirror.put(readValue)

# Main
if __name__ == '__main__':
  # load the fsm
  l = loader()
  l.load(exampleFsm, "myFirstFsm")

  # start execution
  l.start()
```

Listing 1: Example of a FSM implementation.

The FSM is implemented by defining a class which derives from `fsmBase`. In the constructor the user can connect to any amount of PVs, addressing them with their name and then specify the first state to be executed. After that, the example shows the definition of two states, called *idle* and *mirroring*. These are implemented as methods of the class, with the `_eval` suffix in their name. Initially the FSM evaluates the idle state, until the enable PV rises from 0 to 1, which causes the FSM to execute a transition to the mirroring state. In this state the enable PV is continuously checked, so that when an external user revert it back to 0 the FSM goes back to the idle state. Until then, the mirroring state listens to change events to the counter PV and copies the new value to the mirror one. This is done with the `changing()` method to avoid repeating continuously the put operation with the same value, which could slow down the Channel Access, and instead execute a put only once when the counter value changes.

At the end of the example, the loader usage is demonstrated. In this simple case a single FSM is loaded, with just its name as parameter, and the execution is started with the `start()` method. This is by default a blocking call and the execution can be halted using a keyboard interrupt.

CONCLUSIONS AND FUTURE IMPROVEMENTS

Pysmlib is now a stable project and aims to provide a robust solution to the EPICS community to develop modern Finite State Machines. The focus on simplicity and the choice of the Python language make it a good choice for both new and advanced users, who may take advantage of the vast amount of Python scientific libraries.

While the user interface can be considered stable, future updates will focus on improving the internal components to improve the quality and expandability of the code. For example, the current logger facility should be deprecated in favor of the Python logging module to gain all of its functionalities. A more object-oriented approach to events could help to introduce more easily different kind of external events, and thus different kinds of inputs. Currently only the Channel Access inputs are supported, but an expansion to the PV Access could be helpful to users of this newer EPICS protocol. The library could also be generalized to work as a generic Finite State Machine engine, where the user is able to define its custom inputs and connect them to any kind of external system. For this to work, a rigorous I/O interface should be defined and documented.

Other improvements could be evaluated following user suggestions, based on real-world use cases. Contributions and fixes are welcome.

REFERENCES

[1] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal. “EPICS architecture”, in *Proc. ICALEPCS’91*, 1991.

[2] M. Newville, *et al.*, pyepics/pyepics Zenodo. `doi:10.5281/zenodo.592027`

[3] CS-Studio (Phoebus), `https://controlssoftware.sns.ornl.gov/css_phoebus/`.

[4] W. Duckitt, J.K. Abraham, “React Automation Studio: A New Face to Control Large Scientific Equipment”, in *Proc.*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Cyclotrons'19, Cape Town, South Africa, Sep. 2019, pp. 285-288. doi:10.18429/JACoW-Cyclotrons2019-THA03

[5] M. Shankar, M. Davidsaver, M. Konrad and L. Li, "The EPICS Archiver Appliance", in *Proc. 15th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2015)*, Melbourne, Australia, October 17-23, 2015. doi:10.18429/JACoW-ICALEPCS2015-WEPGF030

[6] Bluesky Project and Ophyd, https://blueskyproject.io/.

[7] Krekel *et al.*, pytest 6.0.1, 2004, https://github.com/pytest-dev/pytest

[8] Nox, flexible test automation for Python, https://nox.thea.codes/en/stable/.

[9] Gitlab CI, https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/.

[10] : D. Marcato *et al.*, Pysmlib Github repository, https://github.com/darcato/pysmlib

[11] : D.Marcato *et al.*, Pysmlib on PyPI software repository, https://pypi.org/project/pysmlib/.

[12] B. Warner *et al.*, The Versioneer, https://github.com/python-versioneer/python-versioneer

[13] : D. Marcato *et al.*, Pysmlib online documentation, https://darcato.github.io/pysmlib/docs/html/,

[14] D. Bortolato *et al.*, "New LLRF control system at LNL", in *2016 IEEE-NPSS Real Time Conference (RT)*, 2016, pp. 1-8. doi:10.1109/RTC.2016.7543105

[15] V. Martinelli, *et al.*, "High Level Software for Beam 6D Phase Space Characterization", in *Proceedings of the 10th International Particle Accelerator Conference (IPAC2019)*, Melbourne, Australia, 2019. doi:10.18429/JACoW-IPAC2019-WEPGW025

[16] D. Marcato, Telegram EPICS bot, https://github.com/darcato/telegram-epics-bot

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

NOMINAL DEVICE SUPPORT (NDSv3) AS A SOFTWARE FRAMEWORK FOR MEASUREMENT SYSTEMS IN DIAGNOSTICS*

R. Lange†, ITER Organization, St. Paul lez Durance, France
M. Astrain, V. Costa, D. Rivilla, M. Ruiz, Grupo de Investigación en Instrumentación y Acústica Aplicada, Universidad Politécnica de Madrid, Madrid, Spain
J. Moreno, D. Sanz, GMV Aerospace and Defence, Tres Cantos, Spain

Abstract

Software integration of diverse data acquisition and timing hardware devices in diagnostics applications is very challenging. While the implementation should manage multiple hardware devices from different manufacturers providing different applications program interfaces (APIs), scientists would rather focus on the high-level configuration, using their specific environments such as Experimental Physics and Industrial Control System (EPICS), Tango, the ITER Real-Time Framework or the MARTe2 middleware.

The Nominal Device Support (NDSv3) C++ framework, conceived by Cosylab and under development at ITER for use in its diagnostic applications, uses a layered approach, abstracting specific hardware device APIs as well as the interface to control systems and real-time applications.

ITER CODAC and its partners have developed NDS device drivers using both PCI express extension for instrumentation (PXIe) and Micro Telecommunications Computing Architecture (MTCA) platforms for multifunction data acquisition (DAQ) devices, timing cards and field-programmable gate array (FPGA) based solutions. In addition, the concept of an NDS-System encapsulates a complex structure of multiple NDS device drivers, combining functions of the different low-level devices and collecting all system-specific logic, separating it from generic device driver code.

INTRODUCTION

The Instrumentation and Control Systems (I&C) used in big science facilities (BSF) are based on the use of multi-tier software applications. For example, advanced DAQ and timing systems include complex hardware elements that need software elements to configure all their functionalities. In the last years, the use of field-programmable gate arrays (FPGAs), System on a Chip circuits (SoC) and new development tools have demonstrated that software is a key part of implementing these systems [1]. The key points of using software in advanced I&C systems are adaptability, reusability and maintainability over the entire BSF project lifetime.

The Nominal Device Support (NDS) software framework has been implemented to meet these three goals. Initially developed by Cosylab [2], it was recently improved and extended by the ITER Organization, working with Universidad Politécnica de Madrid and GMV Aerospace and Defence. NDS is a driver development software framework for diagnostics measurement systems [3], focusing on data acquisition and timing devices. NDS device drivers are instantiated and configured to build complex systems, designed to solve specific applications. The applied methodology simplifies code reusability and testability, achieving high levels of software quality. Doxygen documentation, automated tests and static code analysis are used in all NDS modules. Specifically, the NDS framework provides a simplified solution for device driver development in I&C systems that use the Experimental Physics and Industrial Control System (EPICS) [4-6].

NDS SOFTWARE LAYERS

Figure 1 shows the basic structure of a device driver in the NDSv3 framework. The application, called "control system", uses the generic NDS control system interface to communicate with NDS device drivers and extensions. The NDS device drivers use the base and helper classes from the NDS-core library to access the hardware through the operating system's low level drivers.

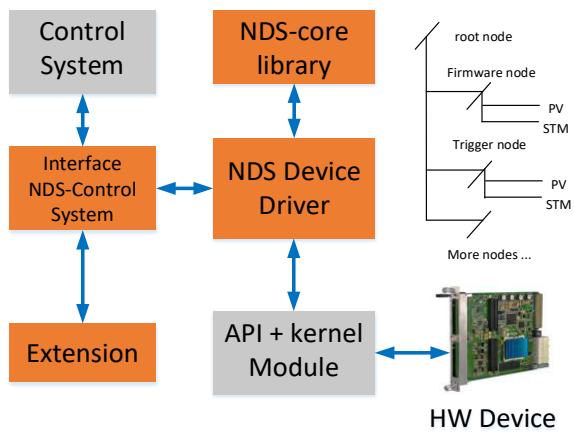


Figure 1: NDSv3 framework elements and basic layers.

NDS-Core

The NDS-core layer (NDS-core library) provides a collection of C++ classes and helpers that standardise and simplify the implementation of the software device driver (NDS device driver) for a specific hardware device or communication interface.

* This work was supported in part by the Spanish Ministry of Science, Innovation, and Universities Projects under Grant ENE2015-64914-C3-3-R and Grant PID2019-108377RB-C33, in part by the Spanish Ministry of Science, Innovation, and Universities Mobility under Grant PRX19/00449, and in part by the Comunidad de Madrid under Grant PEJD-2018-PRE/TIC-8571 and Grant PEJ-2019-AI/TIC-14507.

† ralph.lange@iter.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A device driver implemented with NDS-core can be used by different kinds of applications called "control systems". Examples for such applications are EPICS and the Real-Time Framework developed by ITER [7].

The basic objects of an NDS driver are *nodes* and *process variables (PVs)*. As with previous versions of NDS (see NDSv2 [2]), every variable has a unique name. The structure of this name represents the variable's place in the facility, system, or subsystem. In NDSv3 this concept is maintained, but through abstraction much of the complexity is hidden from the end user. NDS PVs are named data or structures of data relevant to the experiment. To organise them, they are appended to a named node. Subsequently, nodes can be appended to other nodes, creating a tree-like structure that represents the device.

For example, a device might have Analog Input and Analog Output functionalities; these would be appended to a root node or port node to generate the following names: device(node)-ai(node)-ch0(node)-data(PV); device(node)-ao(node)-ch0(node)-data(PV).

Note that two different hardware devices manufactured by different companies implementing the same or similar functionalities can have an NDS device driver with a big part of the code in common. The most crucial point is that they have the same NDS PVs: if these match in name and functionality; software management and device interchangeability are simplified. This is one of the key concepts in the design of NDS based systems at ITER.

NDS-core provides several node and PV type classes to cover most functionality required in a device:

- *Base*: The base node is just a name holder.
- *Port*: A special node that connects with the NDS control system interface layer (see below). The names of PVs are generated from the ports they are appended to.
- *State Machine*: A node with a particular set of functions implementing finite state machine behaviour. The state machine node has a generic set of PVs that control the transitions or propagate states to other state machines below to form hierarchical state machines.

The NDS-core library allows the creation of different NDS PV types. NDS PVs can store a variable or trigger a specific function call, named *delegate*. The read/write operation of an NDS PV allows to read/write the data or trigger the delegate function. The assigned direction, i.e., the definition as an "input" or "output", is determined from the control system's point of view. The NDS PV types (classes) are:

- *Variable In*: This PV contains a value obtained from the hardware device to be sent to the control system. There are two different methods to deliver the data: The control system executes a read operation, accessing the scalar or array value, or the device driver executes a push operation, creating an interruption to the control system.
- *Delegate In*: This PV does not have an attribute, but calls a method when executing a read operation (defined in NDS as a *delegate method*). Typically, this method operates on the hardware through the specific system device driver to read a value.
- *Variable Out* and *Delegate Out*: These PVs contain values received from the control system to be used by the device driver or support the trigger of a function that performs a write operation into the system device driver.

The NDS drivers built in this way are able to communicate using the NDS control system interface layer. This layer has to be implemented once for each control system that needs to use NDS and can be used with all NDS devices drivers.

To control the instantiation of a device and ensure uniqueness of PV naming, the NDS-core layer manages the creation and destruction of device driver instances using a factory (following a singleton pattern implementation). Through the factory, it is possible to get the information about registered drivers and instances. In addition, it gives access to the driver's nodes (hierarchical), the NDS PVs and the state machines.

An important functionality of the NDS-core is the subscription and replication of PVs. When two NDS PVs are connected with a subscription (only possible from an NDS PV In to an NDS PV Out), pushing a value in the NDS PV In generates an update in the NDS PV Out. Similarly, two NDS PV In can be connected by a replication, where pushing a value in one NDS PV In replicates its value to the other. Note that both mechanisms, subscription and replication, are unidirectional. They allow sharing of information between drivers and nodes without the intervention of the control system, which is crucial for nodes related to archiving and real-time communication.

The NDS-core software module consists of:

- The source code implemented in C++ 11.
- Test code implemented with the help of Google-Test [8]. This test code includes a basic implementation of an NDS "control system" (see below) for test and debugging purposes.
- Doxygen [9] support files.

The NDS-core layer does not depend on other software modules (particularly from the EPICS project) and uses the C++ Standard Library. NDS-core can be built on Linux and Windows operating systems.

Control System Interface

The control system interface of the NDSv3 framework implements the interface to communicate with the application that runs the NDS device. Currently, the NDSv3 framework contains two interfaces: one exclusively for running tests of the implemented device drivers (the test control system mentioned) and one to interface with the EPICS control system. The NDS-EPICS control system interface module connects EPICS process database records to NDS PVs. NDS-EPICS is based on the use of the asynDriver module [10, 11], which implements the EPICS Device Support for different records using standardised interfaces. Using NDS-EPICS, the user only needs to define the record templates and the substitutions files. (Examples

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

contained in the software unit.) Creating the interface to a hardware device in EPICS is straightforward if the NDS device driver provides such EPICS templates.

A third implementation of the control system interface, for the TANGO control system toolkit [12], exists in the original NDSv3 development by Cosylab, but has not been used or tested in the context of the ITER developments.

The NDS-EPICS software module consists of:

- The source code of the NDS-EPICS interface.
- An application generating an EPICS Input Output Controller (IOC).
- Test code, implemented using Python with the help of the pyEpics library. These tests verify all implemented asynDriver interfaces.
- Doxygen [9] support files.

DEVICE DRIVERS

The NDS device drivers are plugin libraries. Changing or adding a device drivers does not require to change or recompile the core modules of NDS.

Adding a Device Driver

The development of an NDS device driver using the NDS-core library requires the following steps:

- Analysis of the functional blocks available in the hardware device. This often means a mapping of the Application Program Interface (API) functions provided by the manufacturer in charge of managing the different hardware parts.
- Definition of the hierarchical driver organisation using a tree-like structure of nodes. These nodes can be new classes created by the user or complex nodes from the NDS-core library, e.g., DAQ, Waveform Generation (WFG), Digital Input Output (DIO), Trigger and Clock, Routing, Timestamp or Future Time Event.
- Definition of the operations to be executed in the state machine transitions available in each node. NDS-core defines the states UNKNOWN, INITIALIZING, OFF, ON, RUNNING and FAULT.
- Implementation of the driver constructor. This code includes the initialisation of the resources needed and the creation of the driver hierarchy. In addition, creating the nodes (defined by the user or from the NDS-core library) requires implementing the methods associated with the state machine transitions and the getters/setters of the NDS PVs of type delegate in and out.
- Completion of the methods that call the API of the low level driver for the specific hardware.
- Implementation of the test code, using the test control system and the GoogleTest library.
- Source code documentation using Doxygen.
- Generation of the packages for driver distribution.

The steps to add the interface with EPICS using the NDS-EPICS module are:

- Creation of a software unit including an EPICS application and an EPICS IOC. This application needs to be compiled/linked against the NDSv3 libraries (nds-core, nds-epics) and the asynDriver module.
- Development of the specific EPICS database templates using the examples included in the NDS-EPICS module. Creation of the EPICS substitutions files.
- Configuration of the IOC using the st.cmd file, instantiating the driver and loading the databases.
- Verification of the IOC through an Operator Interface (OPI) or Channel Access operations.
- The NDS-EPICS layer offers the possibility to easily extend the driver implementation with functionalities specific to the EPICS Control System (e.g. execution of init or exit hooks)..

Existing Device Drivers

Table 1 shows a set of device drivers for the PXIe and MTCA platforms that have been developed at different facilities. Functionally, they can be split into four different groups:

- Timing cards (using the PTP standard IEE1588)
- Multi-function DAQ cards
- High sampling rate DAQ cards
- FPGA-based DAQ cards (including a customisable platform).

Of the two solutions in the last group, one is using the FlexRIO device from National Instruments (NI) that can be configured using LabVIEW/FPGA, and the other is based on a device from Advanced Industrial Electronic Systems (AIES), using Hardware Description Language (HDL) with XILINX Vivado.

Table 1: NDS Device Drivers

Devices	PXIe	MTCA
Timing IEEE1588	NI PXI6683H	PTM1588
DAQ	NI X-Series NI PXIe636x	Teledyn ADQ14 MFMC-FMC168 STRUCK SIS300
DAQ-FPGA	FlexRIO (LV FPGA)	MFMC

All device drivers packages contain the driver source code (see Figure 2), Doxygen-based generated documentation, test code developed with GoogleTest, a fully configured EPICS IOC application and Python code testing the EPICS IOC and OPI panels to use the driver using Control System Studio [13].

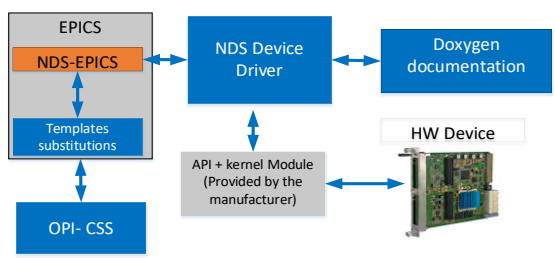


Figure 2: Elements in an NDS Driver software package.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Normally, such NDS device driver modules do not contain information about any specific use or application. They are generic drivers, meant to be integrated into bigger systems that implement specific use cases and include the specific knowledge. This is achieved by combining different driver modules using the extensions introduced in the next section.

EXTENSIONS

Expanding the basic NDS concepts, the ITER Organization and its partners have contributed to the NDS open-source project by improving the existing and providing additional extensions.

NDS Complex Nodes

The main functionalities required to create diagnostics and control instruments were identified and converted into pre-defined functional NDS nodes, called NDS complex nodes. With the intention to homogenise the driver development, the NDS-core library contains nodes supporting common functionalities: DAQ, WFG, DIO, Timing, Triggering, Clock and Routing. In these nodes, the PVs already have defined names, identifying their specific functionality. Therefore, an NDS device driver is in charge of implementing the configuration, status management and data acquisition with the help of software tiers provided by the manufacturer (kernel module and user space API) and the NDS-core library. Additionally, the complex nodes provide an API to simplify calls to update PV values and timestamps. Thus, while the set of PVs provided by the complex nodes is fixed, the final driver implementation can add and manage more PVs for additional functionality.

NDS Plugins

The implementation of a specific application requires the use of multiple device drivers as well as other communication interfaces. In the case of the ITER experiment, there are two such interfaces: the synchronous data bus network (SDN) is oriented to data interchange using a network with a controlled latency and the data archiving network (DAN) is used to stream high throughput data to the archiving engines.

Figure 3 shows the basic architecture for the NDS interfaces to both DAN and SDN.

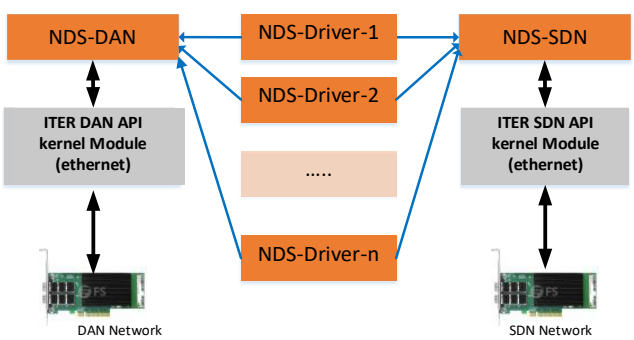


Figure 3: Plugins in an NDS-based application. The blue arrows show the NDS subscriptions.

In order to simplify the use of both interfaces for the developer, the NDS framework defines the concept of an NDS plugin. An NDS plugin is an NDS device driver that requires configuration by the user with the specific parameters of the user's application. This configuration is done through a file that contains all the configuration details in extended mark-up language (XML).

For the SDN plugin, the XML file contains the identification of the SDN topic with its data organisation as well as the mechanism of when to publish the topic. The topic can contain scalar or array fields (of different data types) and structures of these. Each field in the topic is connected to an NDS PV (using the subscription method). The topic is published either when a field changes or periodically with a publishing period defined in the XML file.

For the DAN plugin, the XML configuration similarly contains the details of the NDS PVs (from device drivers) that provide the values (scalar or arrays) that need to be archived.

Besides these two plugins, there is an additional one that implements the access to the EPICS pvAccess protocol. This solution provides a method for NDS based drivers to access industrial or legacy equipment that is interfaced using EPICS IOCs with standard drivers or other applications like middle layer services that provide a pvAccess interface. All EPICS PVs published by these IOCs or software applications can be accessed by NDS applications for reading, writing, and monitoring. This plugin is named NDS-PVXS because it uses the library developed by the PVXS Project [14].

NDS Systems

Developing a diagnostic or an instrument in a big science facility requires integrating multiple hardware devices using different software device drivers. The application developer needs to orchestrate all these hardware elements to implement a correct sequence of operations, i.e., trigger configuration, data acquisition, time stamping, archiving, etc.

In the NDS framework, we have defined the concept of an NDS system, which is a special NDS device driver that manages hierarchically other NDS device drivers (that have already been implemented and tested). This is a key concept, because the developer does not need to program or change anything in the existing NDS device drivers but only configures them for their application. Using this approach, the developer is merely a user of the NDS device drivers and NDS plugins, responsible for configuring and coordinating their use. This noticeably reduces the development time and simplifies the integration of the final solution.

An NDS system is implemented for a specific use case solving the specific requirements. It is not a generic application, and it can be designed to only expose the specific NDS PVs needed, reducing the number of PVs that need to be managed by the control system.

Figure 4 shows the block diagram of an NDS system. A set of C++ classes for the different device drivers implement the subscriptions and replications of NDS PVs to

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

manage an NDS device driver from the NDS system level. The shown structure is the diagram of the system that has been developed as a working example for the NDS system extension.

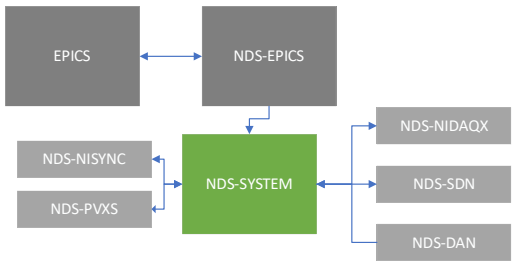


Figure 4: NDS device drivers and NDS plugins in an NDS system.

ADVANCED FPGA-BASED SYSTEMS

The use of advanced FPGA-based DAQ devices, which implement functionalities that can be customised, requires software tools that simplify development and integration. For such applications, the existing NDS framework provides two different solutions.

The first one is based on National Instruments FlexRIO technology, which includes an FPGA that can be configured using the LabVIEW/FPGA tool. In addition, the IRIO Project [15] provides a library that simplifies the integration of these devices. On top of that library, NDS implements an NDS device driver named NDS-IRIO. To implement a custom algorithm in the FPGA using this set of tools, the developer would follow the IRIO design methodology, which automatically connects the FPGA code to the application and generates PVs with the names of the LabVIEW variables, requiring no modifications of the driver code for different use cases.

The second approach is based on the use of the new FPGA, SoC and Adaptive Compute Acceleration Platform (ACAP) with design methodologies that use High Level Synthesis and the OpenCL standard. The NDS-IRIO-OpenCL module implements a software layer that simplifies the integration of such systems [16].

SOFTWARE QUALITY

The software modules implemented in the NDS framework have been developed using the Software Quality model implemented at ITER CODAC. The model is based on the ISO/IEC 15288 standard and there are documents for the different elements: the Software Requirements Specification (SRS), the Software Architecture and Design Description (SADD), the Software Test Plan (STP), the Software Test Report (STR) and the Software User Manual (SUM).

The most meaningful data are the results of the static code analysis and the test coverage of the different modules, obtained with the lcov software utility and a SonarQube [17] installation. The average test coverage value across the different modules is around 70% and has been increased with every new release. Static analysis shows no issues; the cumulated number of reported code smell problems is less than 1%.

CONCLUSIONS

The NDSv3 framework provides a complete set of software modules and documentation to develop complex diagnostics solutions involving timing, data acquisition and other advanced communication interfaces. The NDS implementation model is based on the use of specific device drivers and configurable plugins. The driver developer and the diagnostician can work separately, increasing efficiency and reliability by separating the software layers that contain device specific and application specific knowledge.

The ITER NDS complex nodes standardise most of the PV names, easing the usage of different hardware devices, which should share most PVs. As an added benefit, porting the device driver from one manufacturer API to another should re-use more than 80% of the code.

The new generation of NDS drivers makes use of the different plugins interchangeably and is able to connect to multiple control systems and other software applications. The extensively tested drivers are reaching a very respectable percentage of verification following ITER guidelines, paving the way to a successful deployment in the facility.

PROJECT CONTRIBUTORS

NDSv3 was conceived and originally implemented by Cosylab.

The institutions involved in the recent developments are the ITER Organization, Universidad Politécnica de Madrid (grupo de Investigación en Instrumentación y Acústica Aplicada), GMV Aerospace and Defence, the European Spallation Source and UKAEA-JET/MAST.

The NDS source code for the initial version of NDSv3 is available at GitHub [18, 19]. The extensions implemented by ITER, the device drivers, the plugins and other contributions are available at the ITER Git server [20].

REFERENCES

[1] M. Astrain *et al.*, "Real-Time Implementation of the Neutron/Gamma Discrimination in an FPGA-Based DAQ MTCA Platform Using a Convolutional Neural Network", *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 2173-2178, Aug. 2021. doi:10.1109/TNS.2021.3090670

[2] V. Isaev, N. Claesson, M. Plesko, and K. Žagar, "EPICS Data Acquisition Device Support" in *Proc. 14th Int. Conf on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2013)*, San Francisco, CA, USA, Oct. 2013, paper TUPPC059, pp. 707-709.

[3] J. Vega *et al.*, "New developments at JET in diagnostics, real-time control, data acquisition and information retrieval with potential application to ITER", *Fusion Engineering and Design*, vol. 84, issue 12, pp. 2136-2144, Dec. 2009. doi:10.1016/j.fusengdes.2009.02.055

[4] EPICS Project, https://epics-controls.org

[5] L. R. Dalesio *et al.*, "EPICS 7 Provides Major Enhancements to the EPICS Toolkit", in *Proc. 16th Int. Conf. on Ac-*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR01

celerator and Large Experimental Physics Control Systems (ICALEPCS2017), Barcelona, Spain, Oct. 2017, pp. 22-26. doi:10.18429/JACoW-ICALEPCS2017-MOBPL01

[6] A. N. Johnson *et al.*, "EPICS 7 Core Status Report", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, New York, NY, USA, Oct. 2019, pp. 923-927. doi:10.18429/JACoW-ICALEPCS2019-WECPR01

[7] L. Zabeo *et al.*, "Work-flow process from simulation to operation for the Plasma Control System for the ITER first plasma", *Fusion Engineering and Design*, vol. 146, part B, pp. 1446-1449, 2019. doi:10.1016/j.fusengdes.2019.02.101

[8] GoogleTest/GoogleMock, https://google.github.io/googletest

[9] Doxygen, https://www.doxygen.nl/index.html

[10] M. R. Kraimer *et al.*, "EPICS: Asynchronous Driver Support", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2005)*, Geneva, Switzerland, Oct. 2005, paper PO2.074-5.

[11] asynDriver, https://epics-modules.github.io/master/asyn

[12] Tango Controls, https://www.tango-controls.org

[13] Control System Studio, https://controlsystemstudio.org

[14] PVXS PVA client/server library, https://mdavidsaver.github.io/pvxs

[15] M. Ruiz *et al.*, "IRIO technology: Developing applications for advanced DAQ systems using FPGAs", *2016 IEEE-NPSS Real Time Conference (RT)*, 2016, pp. 1-5, doi:10.1109/RTC.2016.7543090

[16] M. Astrain *et al.*, "A methodology to standardize the development of FPGA-based high-performance DAQ and processing systems using OpenCL", *Fusion Engineering and Design*, vol. 155, 2020, 111561, doi:10.1016/j.fusengdes.2020.111561

[17] SonarQube, https://www.sonarqube.org

[18] NDSv3-core on GitHub, https://github.com/Cosylab/nds3

[19] NDSv3-EPICS on GitHub, https://github.com/Cosylab/nds3_epics

[20] ITER Git Repositories (account required), https://git.iter.org/projects/CCS

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR02

DESIGN PATTERNS FOR THE SKA CONTROL SYSTEM

S. Vrcic, SKA Observatory, Manchester, UK

Abstract

Square Kilometre Array Observatory (SKAO) comprises two radio-telescopes: the SKA Low Frequency Telescope, located in the Murchison Region, Western Australia, with the observing range 50 – 350 MHz; the SKA Mid Frequency Telescope, located in the Karoo Region, South Africa, with the observing range 350 MHz – 15 GHz. The SKA Global Headquarters is in the Jodrell Bank Observatory, near Manchester, UK. The SKA Low and SKA Mid Telescopes use different receptors (log-periodic antennas vs offset-Gregorian dishes), otherwise the telescopes share the same design concept, the design of many components, in particular software, is common to both Telescopes. Work on construction, which officially started on the 1. July 2021, was preceded by the extensive design effort which culminated in successful completion of the Critical Design Review (CDR). This paper describes the design patterns defined so far for the implementation of the Telescope Control System (and applies for both Telescopes).

INTRODUCTION

The Square Kilometre Array Observatory (SKAO) is an International Organisation that comprises two radio-telescopes and spans three continents:

- The Global Headquarters (HQ) is located at the Jodrell Bank Observatory near Manchester, UK.
- The SKA Low Telescope, located in Murchison Region in Western Australia operates in the frequency range 50 – 350 MHz.
- The SKA Mid Telescope, located in the Karoo region in South Africa, operates in the frequency range 350 MHz to 15 GHz. The SKA Mid observing range is divided in 6 bands; each receiver can collect and process data for one band at a time; instantaneous bandwidth varies from band to band.

The SKAO was instigated by an idea to build a radio-telescope with a collecting area of 1 km^2. The science goals are described on the SKAO website [1].

Instead of building a single gigantic dish, radio-astronomers use a technique called interferometry. A Radio-Interferometer superimposes electromagnetic waves collected by multiple receivers to amplify the signal of interest and eliminate signals generated by the ground-based sources and by other man-made equipment (including the telescope itself). The signals generated by sources not of interest are generally referred to as radio-interference (RFI). The SKA Telescopes are interferometers.

The SKA Low Telescope comprises 131,072 log periodic antennas, organised in 512 stations. The stations are placed so that the Telescope consist of the densely populated core and three spiral arms with receding density. Signal collected by the antennas is digitized, the beams are formed using input from the antennas the belong to the same station, the beams formed by different stations are aligned in time, and correlated (input from each pair of stations is complex cross-multiplied) to extract information of interest and eliminate RFI. Integration in time and/or frequency may be performed to reduce the amount of output data. Up to this point, data processing is performed in real-time. The output of correlation is then captured, data sets formed and stored for further processing.

The SKA Mid Telescope comprises 197 offset-Gregorian dishes, each with the diameter of 15 meters. To cover the required frequency range, each dish is equipped with several receivers, but only one can be placed at the focus at any given time (consequently, before starting a new experiment a receiver may need to be replaced). The signal captured by the single pixel feed is digitized and processed in the same manner as in the Low Telescope.

After many years of preparation, the construction of the SKA Telescopes officially started on 1. July 2021 (artist impression of the SKA Telescopes shown in Fig. 1).

During the pre-construction phase design were developed and reviewed for each part of the Telescopes, including the Control System. The general principles for the design of the SKA Telescope Control System were established relatively early in the design process. Following the Critical Design Review (CDR), while preparing for transition to construction, work on the detailed design and development of the Control System software started, although with the limited resources. The lessons learned over that period are being applied as we update documentation, improve the development environment, and refine the design, including the design patterns.

This paper provides an overview of the design approach and patterns used in the SKA Control System, with the goal to collect input from the colleagues working on similar projects.

Figure 1: Artist impression of the SKA Telescopes.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

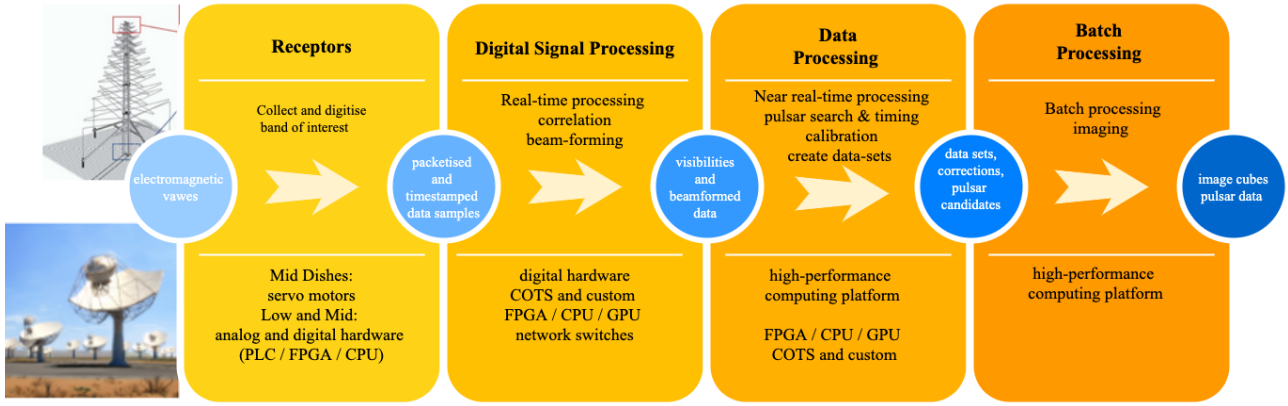


Figure 2: SKA Telescope functional overview and the technologies used.

SKA TELESCOPE CONTROL SYSTEM REQUIREMENTS

Figure 2 provides a high-level overview of the functionality required and technologies used to implement the SKA telescopes. With exception of the antennae, similar technology is used in both telescopes. Analog hardware is used in receivers; once the collected signal is digitized, digital hardware and computers are used for signal processing and to packetize, transmit, receive, and store astronomical data as it is passed from one sub-system to the other. The SKA Telescopes use computing hardware equipped with Field Programmable Gate Arrays (FPGAs), Graphical Processing Units (GPUs) and Central Processing Units (CPUs). The log-periodic antennae in the SKA Low Telescope do not have moving parts. The dishes in the SKA Mid Telescope are equipped with servo motors that move the dish to point the dish focus towards the desired source and rotate the indexer (the tray where receivers are placed) to place one of the receivers in the focus.

The requirements for the Telescope Control System can be classified as: requirements related to monitoring and control of the equipment and requirements related to monitoring and control of the experiments (in the context of SKA referred to as observations).

Equipment Monitoring and Control

Requirements related to monitoring and control of the equipment and software are standard requirements for the similar systems used in industry and scientific facilities. The following list is a brief overview of the required functionality:

- Monitoring (report status, periodically and on request, report state and mode changes, errors, and faults).
- Archiving (periodically report and store the status of all the components and systems of interest).
- Provide interfaces that allow operations to change configurable parameters and trigger state/mode changes. Both Human Machine Interfaces (HMI) and Machine to Machine Interfaces (MMI) are required.
- Support for hardware / software / firmware updates.
- Support for debugging, testing and maintenance.

The above listed requirements apply for all the levels of the monitor and control hierarchy: the telescope, each subsystem, and individual components.

The components and subsystems differ in complexity, as does the number of attributes, configurable parameters and commands exposed by different subsystems and components. For each component and subsystem, it must be determined what is the information of interest and how it should be represented; this will be an on-going activity during the detail design, development, integration, and commissioning.

Graphical User Interfaces (GUIs) that display overall status of each telescope, status of the major subsystems and interfaces and the list of active alarms will be provided. In addition, more detailed 'engineering' views will be provided to be used during the development, integration, testing and commissioning; and will continue to be used in operations by the maintenance staff. The plan is to arrange the views of the telescope hierarchically so that a click on a faulty component opens a more detailed view of the component.

Figure 2 shows the telescope functions and equipment directly involved in capturing and processing astronomical data. Figure 2 does not show functionality and equipment provided by infrastructure, such as: facilities, roads, lending strips, power, water, networking, IT equipment, and masers and other equipment that provides very precise time required for telescope operations. Most of the infrastructure comes with own COTS monitor and control systems, and User Interfaces (operator and maintainer consoles). Also not shown are Data Archives: Observing Data Archive, Scientific Data Archive, and Engineering Data Archive. The Telescope Control System will communicate with the COTS systems to collect information of interest (overall status, significant events and alarms), but will not duplicate all information. Also not shown are Data Archives: Observing Data Archive, Scientific Data Archive, and Engineering Data Archive.

Observation Monitoring and Control

Observing Modes Both SKA telescopes support multiple observing modes: imaging, pulsar search, pulsar timing, and VLBI beam-forming, and several sub-modes for each observing mode. In addition, to support exploration of the transients, both telescopes are able to capture and, on re-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR02

quest, offload a limited amount of captured raw data for each receptor. The Telescope Control System API (Application Programming Interface) shall allow users to configure a telescope for the desired observing mode and take advantage of the telescope versatility; in other words, the Control System shall not restrict users to a limited number of pre-defined configuration sets.

Concurrency and Commensality The SKA Telescopes provide processing resources for concurrent processing in all 4 observing modes; thus, allowing search for pulsars and precise timing of the known pulsars to be executed concurrently with surveys and imaging observations. The Mid Telescope, due to a huge instantaneous bandwidth (5 GHz), provides a limited concurrency for Band 5 observations.

Subarrays Not all science observations require all the receptors, some require only the receptors in the core while others are mostly interested in long baselines and therefore use only a small number of receptors in the core. To provide for efficient use of observing time, both telescopes are required to allow operations to subdivide the array (receptors and processing resources) and operate each subarray as an independent telescope, in terms of the observing mode, and the start and stop time of the observations. The requirement to support subarraying provides for more efficient use of resources but increases complexity of the Control System (and other subsystems).

Observation Monitor and Control – Key Functions The following is a brief overview of the Telescope Control System functions related to observation monitor and control:

- Assign resources (receivers and processing resources) to a subarray.
- Configure subarray for the observing modes. A number of parameters can be specified for each observing mode; a complete sub-array configuration is passed in the form of ASCII (American Standard Code for Information Interchange) encoded JSON (JavaScript Object Notation) script.
- Point the Mid Telescope dishes, track sources.
- Provide regular updates for delay and phase tracking corrections (to be applied during signal processing).
- During the observation, pass the updates received from the Science Data Processor (SDP) to the components used by the subarray, these include parameters for pointing and steering beams (station beams in the Low Telescope; beams for pulsar search, pulsar timing and VLBI in both telescopes), antennae weights to be applied in beamforming, RFI extinction related parameters and more.
- Start and stop tracking and data processing.
- Store data products (data sets).
- Release resources from a subarray.
- Derive and report overall status of the sub-arrays.
- Keep track of resource status and allocation, report status changes periodically and as requested.

Non-Functional Requirements (NFRs)

The key non-functional requirements for the Telescope Control System are:

- **Resiliency** – ability to function in presence of errors and faults, and to recover from errors and faults. Resiliency is extremely important for the Control System, which should be able to report status and accept commands even when much of the telescope equipment is only party functional, non-responsive or failed. Control System must be able to continue functioning when some of the components are unresponsive or provide unexpected response, and to resume normal operations when previously non-responsive or failed components recover.
- **Reliability** – Control System must be reliable, it should be the last component to fail. In all circumstances, the Control System should be able to reliably monitor and report status of other components, execute requests and commands and raise alarms when needed.
- **Performance** – Control System shall execute requests and detect and report status changes, errors, and alarms in timely manner. (The exact meaning of 'timely manner' is quantified in the actual requirements).

The key Control System related design decisions are to a great extent influenced by the concerns related to resiliency, reliability, and performance. Development of the SKA Telescope Control System is still in the early stages, we still have to prove that the design choices result in implementation that meets the requirements.

SKA TELESCOPE CONTROL SYSTEM DESIGN

This paper summarizes key design decisions and provides an overview of the design patterns for the implementation of the SKA Control System.

Distributed Control Logic

Each SKA Telescope consists of several major subsystems. Receptors (dishes in the SKA Mid Telescope, antennas/stations in the SKA Low Telescope), Central Signal Processor (CSP) and Science Data Processor (SDP) capture and process astronomical data. Other sub-systems provide infrastructure, communication networks, very precise timing, and preparation and execution of observations and telescope monitor and control.

Given the number of receptors and the signal and data processing capacity required, each sub-system consists of hundreds of hardware and software components. To handle complexity and size of the system, the sub-systems are further decomposed in smaller sub-systems and products.

Logic required for monitor and control is embedded in each sub-system and hierarchically organized. Each 'level' performs aggregation and reports overall status of the subsystem or component. Configuration and commands are passed from top to bottom, at each 'level' the higher-level parameters are translated into detailed configuration of the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

subordinate components. This approach reduces complexity of individual software components and enables scalability and modifiability – important quality attributes as the SKA Observatory is expected to be used for at least 50 years; during that time refresh of the digital hardware and computing platforms is expected, as well as expansion of the array and addition of new observing and/or processing modes.

Physical vs Functional View

In an interferometer, the functional view does not always directly map to the physical view. The requirement to operate each sub-array as an independent telescope further complicates 'mapping' of the functionality to physical equipment. SKA Control System provides two views:

- Physical (equipment and components).
- Functional (subarrays, resources, and capabilities).

SKA Base Classes

A set of the SKA Base Classes has been developed to be used as the base for development of the monitor and control software. All components and sub-systems implement the same set of state and mode indicators, and commands to trigger state transitions. Each sub-system is required to implement a Master Controller which performs aggregation of the status and coordinates execution of commands. The Maser Controller:

- Monitors status of all components and based on the collected information derives overall status of the sub-system.
- Implements a set of commands that can be used to trigger state transitions, update software (and firmware), set logging level and more.

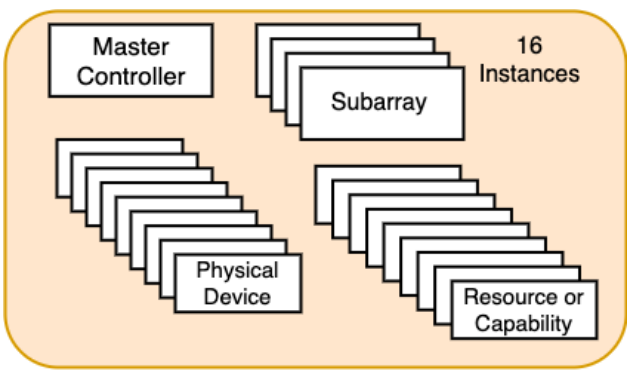


Figure 3: Design Pattern for implementation of monitor and control for sub-systems that process astronomical data.

Figure 3 shows the design pattern for monitor and control of sub-systems on the signal chain. Each sub-system implements:

- Master Controller which reports overall status of equipment and components, implements commands to trigger state transitions, upgrades, and other housekeeping tasks.
- Monitor and control for all physical devices.
- Multiple instances of Subarray (presently 16), which provides monitor and control functionality related to configuration and execution of observations.
- Monitor and control for resources and capabilities, i.e. entities that implement individual functions (in most cases implemented as FPGA bit streams and software pipelines).

TANGO Controls Framework

Early in the design phase a decision was made to use TANGO Controls [2] as the base for implementation of the SKA Telescope Control System. TANGO Controls is an open-source toolkit that can be used for controlling any kind of software and hardware.

From the design point of view the key concepts are:

- TANGO Device – a software component (instantiation of a TANGO Device Class). Each TANGO Device models a device, software component, or sub-system.
- Tango Device Server – provides an execution environment for one or more TANGO Devices.
- TANGO Data Base – all TANGO Devices register with the TANGO DB which can be used to obtain the addresses of registered devices, and then access devices directly.

TANGO uses CORBA (Common Object Request Broker Architecture) [3] for request-reply communication and ZeroMQ [4] for publish-subscribe.

TANGO supports development in C++, Java and Python. SKA Control System is developed in Python.

Decoupling TANGO API from the System Logic

As already mentioned, the SKA Control System is hierarchically organized and provides the same API at all levels of monitor and control hierarchy. This implies 'deep adoption of TANGO", i.e. TANGO API is provided for all sub-systems and components.

Unwanted consequences of this approach are:

- The framework (TANGO) becomes entangled into every aspect of the control system logic; the Control System becomes overly dependent on TANGO.
- Confusion regarding the significance of the status indicators, i.e. how to distinguish the TANGO Device (software model) from the device (sub-system or software construct) being modeled.

The SKA solution for those issues is to decouple the TANGO API (layer) from the 'system logic', so that the TANGO Device provides only the API, while the logic required for monitoring, status aggregation, error and fault detection, execution of commands, etc. is handled by a non-TANGO class, the so-called Device Manager.

Loose Coupling of Components

In the case of the SKA Telescopes, due to large number of components:

- The FMECA (Failure Modes, Effects and Criticality Analysis) shows that, once the full system is deployed, at any given time some of the components will be out

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR02

of service or undergoing maintenance. This means that Control System must be able to correctly function in presence of errors and failures. The Control System must be able to handle situations where one or more components become unresponsive, and to resume normal operations once the components become available. Also the Control System should not expect individual components to complete state transitions in a particular order, each component may transition to ON, STANDBY or OFF individually, or may become unresponsive.

- Execution of commands at a telescope or sub-system level will take significant time, in some cases tens of seconds. This is particularly true for transitions that increase or decrease power consumption (ON/STANDBY/OFF), where delay is intentionally introduced to control inrush power. Reconfiguration of a subarray observing modes also may require significant time, as the commands propagate through the hierarchy of components, and the FPGAs get reprogrammed.

Although different in nature, the above-described aspects of the telescope behaviour may be addressed using the same technique, or more precisely by using techniques that promote loose coupling of components.

The SKA Control System uses the following techniques:

- Asynchronous communications. The commands that require I/O, must to be forwarded to subordinate components or sub-systems and/or require communication over the network, or access to hardware are implemented as non-blocking, meaning that the originator of the command is not blocked waiting for the command to be executed. The server (TANGO Device) immediately acknowledges receipt of the command and returns command to the caller; during the command execution, the device on request can report the progress, once the command is completed the originator is notified via an event (status change).
- Input queue for commands – When received, a command is added to the end of the input queue; the worker thread executes commands one-by-one. The input queue is implemented to handle exceptional situations, when the system is under stress due to unexpected events; the system will be tuned so that, under normal circumstances, the queue is empty 90% of time. Not all devices have to implement the input queue.
- The subarray observing mode configuration is passed in a form of ASCII encoded JSON script, which contains the complete configuration for a single unit of observing. Use of data-centric self-contained messages and simple common types in data model support loose coupling.

SUMMARY

The following is the summary of the key design decisions and patterns chosen for implementation of the SKA Telescope Control System:

- Hierarchical organization.
- Distributed control of process logic.
- Decouple physical and functional view.
- Set of SKA Base Classes.
- Standard set of state/mode indicators, and commands that trigger state/mode transitions.
- TANGO Controls framework.
- TANGO API provided at all levels of hierarchy.
- Decouple TANGO API from “system logic”.
- Use JSON to pass configuration messages.
- Use asynchronous communications.

Development of the SKA Telescopes is still in early stages, and the team still has to prove that the design choices made so far are adequate for the challenging requirements.

ACKNOWLEDGEMENTS

Design for the SKA Telescope Control System is result of many years of work of many SKA team members, this paper a brief overview of the status.

REFERENCES

[1] SKA Observatory, https://skatelescope.org

[2] Tango Controls, https://www.tango-controls.org/

[3] Common Object Request Broker Architecture™, https://www.corba.org/

[4] ZeroMQ messaging library, https://zeromq.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR03

CONTROL SYSTEM FOR 6 MeV LINEAR ACCELERATOR AT LINAC PROJECT PINSTECH

N. U. Saqib†, M. Ajmal, A. Majid*, D. A. Nawaz‡, F. Sher§, A. Tanvir
LINAC Project, PINSTECH, Islamabad, Pakistan

Abstract

At LINAC Project PINSTECH, 6 MeV electron linear accelerator prototypes are being developed for medical as well as industrial purposes. Control system of the linear accelerators is a distributed control system mainly comprised of EPICS and PLCs. Graphical User Interface (GUI) are developed using Phoebus Control System Studio (CSS) and Siemens WinCC Advanced software. This paper focuses on design, development and implementation of accelerator control system for various subsystems such as RF, vacuum, cooling as well as safety subsystems. The current status of the control system and services is presented.

INTRODUCTION

Particle accelerators are complex machines that have been playing a major role in scientific innovations and discoveries. A particle accelerator is a device that uses electromagnetic waves, in the microwave range, to accelerate charged particles (electrons) to produce high energy charged particle or X-ray beams. In case of electron linear accelerators, electrons are generated from electron source such as electron gun (e-Gun) and are accelerated in a series of accelerating cavities by Radio Frequency (RF) waves, which are fed into the accelerating cavities by an RF source such as a klystron or a magnetron.

LINAC Project at PINSTECH aims at developing indigenous RF linear accelerators for medical and industrial applications. Along with ongoing progressive research and development, prototypes of medical linear accelerator (*Medical Linac*) for radiotherapy application and industrial linear accelerator (*Industrial Linac*) for Non-Destructive Testing (NDT) application are under development. Both of these accelerators are 6 MeV standing wave electron linear accelerators that produce high energy X-rays by striking high energy electron beam onto a tungsten target.

There are several subsystems that are required for operating a linear accelerator: an e-Gun system to produce and extract electrons, an RF system to accelerate the electrons produced by e-Gun, a good vacuum system to avoid arcing or breakdowns, a cooling system to maintain the temperatures of various devices at a desired point for stable operation. As particle accelerators grow more complex, so do the demands placed on their control systems. The complexity of the control system hardware and software reflects the complexity of the machine. Accelerator Controls and Electronics (ACE) Group of LINAC Project is responsible for controls, electronics and IT infrastructure for Medical and Industrial Linac prototypes. Figure 1 shows block diagram of the linear accelerator that contains main subsystems and their interfacing. All these systems need to be controlled and monitored by a sophisticated control system, which enables safe and smooth remote operation of the Linac from control room. Next subsections provide a brief description of Medical and Industrial Linac prototypes.

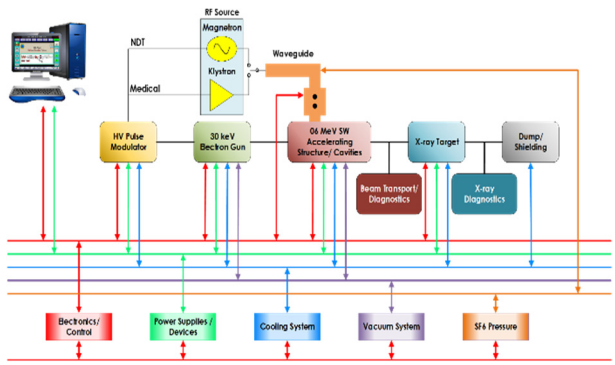


Figure 1: Block Diagram of Linac.

Medical Linac

A medical linear accelerator customizes high energy x-rays or electrons to conform to a tumor's shape and destroy cancer cells while sparing normal tissues. Cancer is a leading cause of death worldwide and its types such as breast, lung, liver, prostate, head and neck carcinoma are most commonly diagnosed in Pakistan. Increasing number of cancer patients has made demand of cancer treatment machines. Therefore, it was need of the time to build an indigenous medical linear accelerator to cope with soaring number of cancer patients in available financial resources. For this purpose, a Klystron based 6 MeV Medical Linac prototype is being developed at LINAC Project PINSTECH. Accelerating cavity was designed indigenously, fabricated and brazed locally. Initially, there was one vacuum port and vacuum didn't sustain to a good pressure, recently a second vacuum port is installed and RF conditioning is under process.

Industrial Linac

Out of several applications of industrial linear accelerators, one is radiography in for Non-Destructive Testing purposes, which provides a method for examination of internal structure and integrity of given specimen/material. Keeping this in view, an indigenous magnetron based 6 MeV Industrial NDT Linac prototype is in final stages at LINAC Project PINSTECH. Accelerating cavity was designed indigenously, fabricated and brazed for this purpose. Most of the parameters of both the linear accelerators are similar as shown in Table 1.

† najm.control@gmail.com
* aaminahmajid@gmail.com
‡ danishplcinc@gmail.com
§ falaksher@gmail.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR03

Table 1: Specifications of the Linac

Parameter	Value
Maximum Energy	6 MeV
Input Power	2.5 MW
Pulse Repetition Frequency	50 Hz
RF Pulse Width	4 µs

CONTROL SYSTEM DESIGN

Control system acts as brain of a linear accelerator. It allows smooth and safe operation of the Linac by providing remote control/monitoring of accelerator hardware as well as interlocked environment for safety and protection. Control system at our facility is EPICS and PLC based four layer architecture [1,2]: Graphical User Interface (GUI) layer consisting of Phoebus Control System Studio (CSS), Middleware layer consisting of EPICS, Front-end layer consisting of PLCs, instruments and controllers, and Accelerator Hardware layer consisting various physical phenomenon that need to be controlled and monitored by the control system. Next subsections describe various components of the control system of our facility [3].

Input/Output Controllers (IOC)

IOCs in our EPICS [4] based control system are soft IOCs deployed on CentOS 7 Linux Virtual Machine. IOCs communicate with various commercial-off-the-shelf instruments and controllers via interfaces like Ethernet, Serial (RS-232/422/485) and GPIB. They send commands and acquire signals from instruments and controllers. The IOCs deployed in our control system are organized in a hierarchical manner. First of all, in folders according to the subsystem they represent (RF, vacuum, cooling) and then according to the device they control (signal generator, delay generator, klystron modulator). For example, an IOC that controls signal generator which provides pilot signal to Klystron pre-amplifier is placed at ${HOME}/epics/ioc/RF/signalGenerator. EPICS support modules provide enhanced functionalities for IOC applications. Support modules currently installed and configure in our control system include Asyn, Stream Device, Modbus, Calc, s7nodave and Sequencer modules. Separate virtual machines are used for IOC development and IOC deployment. At system reboot, all IOCs are started from a bash script containing path to their startup script files.

Programmable Logic Controllers (PLC)

Siemens Simatic S7-300 and S7-1200 programmable logic controllers are installed, configured and programmed to interface field devices. Input/Output (I/O) modules include analog input, analog output, digital input, and digital output modules. Communication modules for Serial RS-232/422/485 interface are also installed for remote control & monitoring of commercial-off-the-shelf equipment. Interfacing between EPICS and Siemens PLCs is achieved via EPICS s7nodave support module that allows interfacing of Siemens Simatic S7 controllers with EPICS. PLCs act as front-end controllers in our system to interface with the accelerator hardware.

Operator Interfaces (OPI)

Operator interfaces (OPI) developed in Phoebus Control System Studio [5] are used to operate the linear accelerators from control room. Previously, OPIs were developed in Eclipse based Control System Studio, but later on, upgraded to Phoebus based Control System Studio. This transition provided more reliability and better features required by the control system. Several OPIs are designed according to the functionality they provide (Control, Monitoring, Trends). Also, specific equipment details OPIs are developed for various equipment such as Klystron Modulator, Oscilloscope etc. Example Phoebus OPIs are shown in Fig. 2.

For Industrial Linac, Human Machine Interface (HMI) screens are developed in Siemens WinCC Advanced and shown in Fig. 3.

Figure 2: Phoebus OPIs.

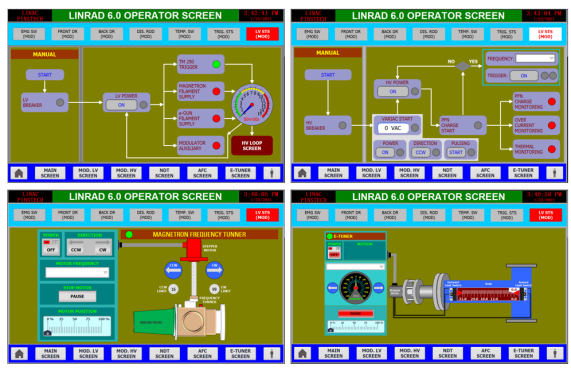


Figure 3: WinCC Advanced HMIs.

Services

An archiver system and an alarm system are necessary services of a control system. These services inform operators about events happening during linear accelerator operation and their history, as well as provide data analysis functionalities to scientists and engineers.

Archiver system saves process variables history in a database which is a necessary task for a control system. It provides lot of benefits such as detection of a system fault, detection and analysis of undesired behavior of the system and also provides data to scientists and engineers for research purposes. Best Ever Archiver Toolkit Yet

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

(BEAUTY) is used for data archiving in our system. It contains archive engine command-line tool, which is provided with configuration and settings from XML files. Archive engine then utilizes Channel Access protocol to access process variables contained in EPICS IOCs into MySQL relational database. Phoebus CSS Data Browser is used to display history trends for the operating personnel. Figure 4 shows architecture of the archiver system.

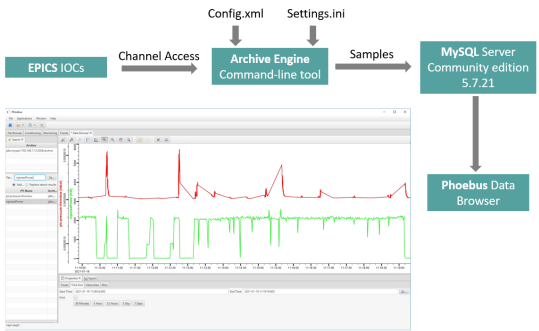


Figure 4: Archiver System Architecture.

Alarm system is a great tool for the operators to be notified about events happening during the operation of accelerator. Alarm system consists of Best Ever Alarm System Toolkit (BEAST) which contains alarm server. Configuration and settings are provided from XML files. Previously, alarm system consisted of RDB & JMS but after update to Apache Kafka, we have updated the alarm system accordingly. Alarm Panel, Alarm Tree and Annunciator provide a great alarm system. Figure 5 shows the alarm system architecture.

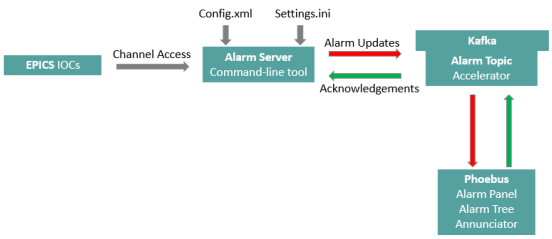


Figure 5: Alarm System Architecture.

Computing Infrastructure & Network

Network deployed at LINAC Project is an Ethernet based Local Area Network (LAN). The overall LAN is divided into two segments: Technical Network and Office Network. Both networks are isolated and contain independent IT infrastructure. Control system consisting of measurement and control devices including commercial-off-the-shelf equipment and related devices are connected to the Technical network while Office network consists of users related devices including scientists' and engineers' PCs, printers, scanners, test equipment etc. Dell EMC PowerEdge servers are configured and installed to provide reliable services for both the networks.

Technical Network contains all components related to control system consisting of servers, desktop computers and commercial-off-the-shelf equipment. This provide isolation for achieving security and reliability. Figure 6 shows various application/services provided by Dell EMC PowerEdge R630 virtualized server deployed in the Technical Network.

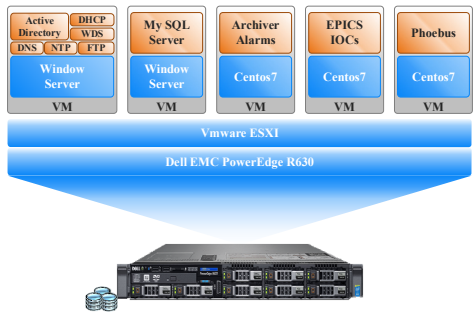


Figure 6: Technical Network Virtualization.

ACCELERATOR SUBSYSTEMS

This section describes various subsystems of linear accelerator and how their components are interfaced with the control system. Subsystems of Medical and Industrial Linacs that significantly differ from each other with respect to control system point of view are described separately in each subsystem's section.

RF System

Medical Linac RF system consists of Klystron as RF source, Scandinova Klystron Modulator as HV source for the Klystron, Stanford Research System Signal Generator as RF pilot signal provider to Klystron preamplifier, BNC Digital Delay Generator as trigger of the modulator, Rohde & Schwarz Oscilloscope as RF Power measurement device. All of these devices are interfaced with EPICS IOCs using Ethernet interfaces. Klystron modulator communicates via Modbus TCP protocol, Delay Generator via Telnet protocol and other instruments via Vxi-11 protocol. Figure 7 shows RF system interfacing with the control system. RF conditioning process is automated using EPICS Sequencer module to minimize human interaction requirement and make the process error prone and reliable.

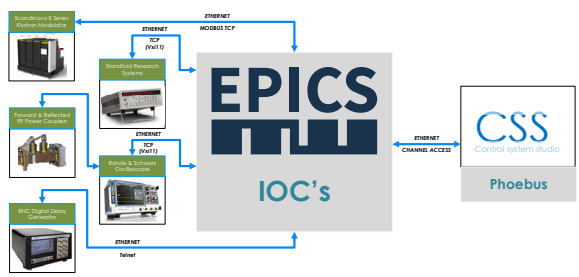


Figure 7: RF System of Medical Linac.

Industrial Linac RF system consists of S-band Magnetron as RF source, indigenously developed 48 kV/110 A HV Pulse Modulator as HV source for the Magnetron. Modulator control system consists of Siemens Simatic S7-1200 PLC CPU along with analog and digital input/output modules. EPICS communicates with S7-300 via s7nodave module and provides data to Control System Studio via

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR03

Channel Access Protocol. Figure 8 shows interfacing of RF system and Control system.

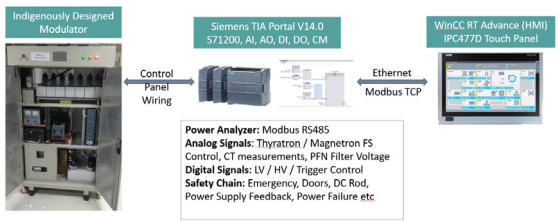


Figure 8: RF System of Industrial Linac.

Electron Gun System

Electron Gun system consists of indigenously developed high voltage pulse modulator to provide HV power to electron gun. Lambda ZUP Power Supply are installed to provide Low Voltage (LV) power to the electron gun filament.

Medical Linac EPICS IOC interfaces ZUP Power Supply via RS-422 and ramp up/ramp down of filament voltage, and conditioning of the filament is automated using EPICS Sequencer module.

Industrial Linac interfaces this supply via RS-422 PLC communication module and ramp up/ramp down of filament voltage, and conditioning of the filament is automated with PLC Ladder Logic programming. Figure 9 shows interfacing of e-Gun system and Control system.

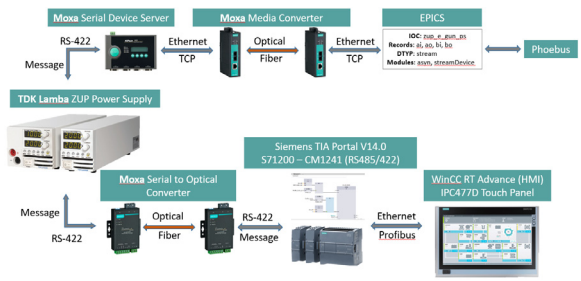


Figure 9: e-Gun System.

Vacuum System

Vacuum is monitored using IMG-300 gauge card of Agilent Technologies XGS-600 vacuum gauge controller. The controller is interfaced with EPICS via RS-232 interface and also with Siemens S7-300 PLC via analog input module. Figure 10 shows interfacing of Vacuum system and Control system.

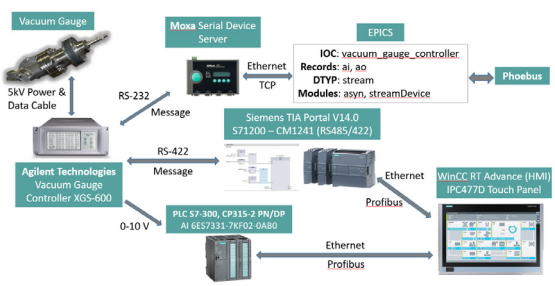


Figure 10: Vacuum System.

Cooling System

Chiller provides cooling to the Linac hardware. K-type thermocouples are used to monitor temperatures at various points along the accelerator. Phoenix temperature transducers are interfaced with S7-300 PLC via analog input module. EPICS communicates with S7-300 via s7nodave module and provides data to Control System Studio via Channel Access Protocol. Figure 11 shows interfacing of Cooling system and Control system.

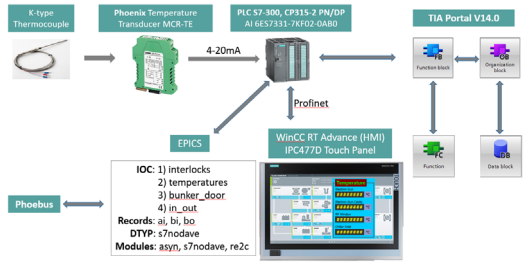


Figure 11: Cooling System.

EXPERIMENTAL SETUP

Accelerator Hall (Main Lab) situated at LINAC Project PINSTECH hosts both Medical Linac experimental setup and Industrial Linac prototype. These linacs are operated from Control Room that is adjacent to Accelerator Hall. Control Room contains six LED screens. We have also configured and installed CCTV system for surveillance. One LED screen is used for CCTV cameras installed at different locations and five screens are used for different operator interfaces. Figure 12 shows the experimental setup of Medical Linac, Fig. 13 shows the NDT Linac prototype and Fig. 14 shows the Control Room.

Figure 12: Medical LINAC Experimental Setup.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 13: NDT LINAC Prototype.

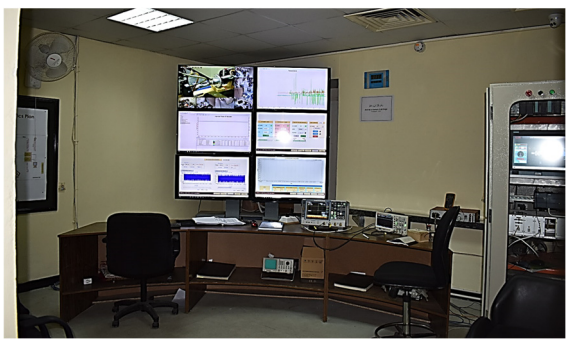


Figure 14: Control Room.

CONCLUSION

Accelerator control system provides smooth, safe and reliable operation of the linear accelerators at the facility. Control system is continuously upgraded according to requirements from operating personnel and other groups of the Project. Accelerator control system community recommended practices are being followed to develop a state-of-the-art control system which can be comparable with control systems of other facilities.

ACKNOWLEDGEMENTS

The authors thank all groups of the LINAC Project for their support. Also, we thank JACoW and ICALEPCS for providing literature that guides us towards right direction.

REFERENCES

[1] T. Wilksen *et al.*, "The Control System for the Linear Accelerator at the European XFEL: Status and First Experiences", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 1-5. doi:10.18429/JACoW-ICALEPCS2017-MOAPL01

[2] Y. B. Yan *et al.*, "The Control Systems of SXFEL and DCLS", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 525-530. doi:10.18429/JACoW-ICALEPCS2017-TUPHA058

[3] H. Kaji *et al.*, "Control System of SuperKEKB", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOAPP01. doi:10.18429/JACoW-ICALEPCS2019-MOAPP01

[4] EPICS, https://www.epics-controls.org

[5] Control System Studio, https://controlsystemstudio.org

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR04

CONTROL SYSTEM OF CRYOMODULE TEST FACILITIES FOR SHINE*

H. Y. Wang, G. H. Chen†, J. F. Chen, J. G. Ding,
M. Li, Y. J. Liu, Q. R. Mi, H. F. Miao, C. L. Yu
Shanghai Advanced Research Institute, Chinese Academy of Sciences
Shanghai 201204, P. R. China

Abstract

Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE) is under construction. The 8 GeV superconducting Linac consists of seventy-five 1.3 GHz and two 3.9 GHz cryomodules. A cryomodule assembling and test workshop is established.Multiple facilities have been built for cryomodule and superconducting cavity test, including two vertical test facilities, two horizontal test facility, one multiple test facility and one liquid helium visualization facility.The local control systems are all based on Yokogawa PLC, which monitor and control the process variables such as temperature, pressure, liquid level and power of the heater. PID and other algorithms are used to keep liquid level and power balance. EPICS is adopted to integrate these facilities along with vacuum devices, solid state amplifiers, LLRF and RF measurement system, etc. The details of the control system design, development and commissioning will be reported in this paper.

OVERVIEW

Owing to the wide range of applications of X-rays in the research fields of physics, chemistry and biology, facilities with the ability to generate X-rays were developed continuously in the last century. The free electron laser (FEL) is a novel light source, producing high-brightness X-ray pulses. To achieve high-intensity and ultra-fast short wavelength radiation, several X-ray FEL facilities have been completed or under construction around the world [1].

The first hard X-ray FEL light source in China, the so-called Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), is under construction. It will utilize a photocathode electron gun combined with the superconducting Linac to produce 8 GeV FEL quality electron beams with 1.003086MHz repetition rate.

CRYOMODULE

Cryomodule (Fig. 1) is the key components of the superconducting linear accelerator, which composes of superconducting cavities, superconducting magnet components, beam position detectors, cryogenic cooling system, vacuum system, and mechanical support system. SHINE requires 75 1.3GHz superconducting cryomodules, which are connected in series to form the accelerator L1, L2, L3, and L4. In addition, two third-harmonic cavities with a frequency of 3.9GHz superconducting modules will be used to linearize the longitudinal emittance of the electron beam [2].

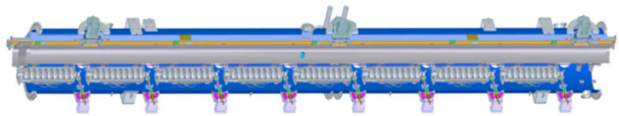


Figure 1: Cryomodule.

CRYOMODULE TEST FACILITY

A superconducting cryomodule workshop is built for SHINE project. The infrastructure(shown in Fig. 2) includes ultra-clean processing and assembly, precision mechanical assembly and testing, cryogenic component multi-functional test facility system, superconducting cavity vertical and horizontal test facility system. They are used for superconducting cryomodule assembly and functional testing.

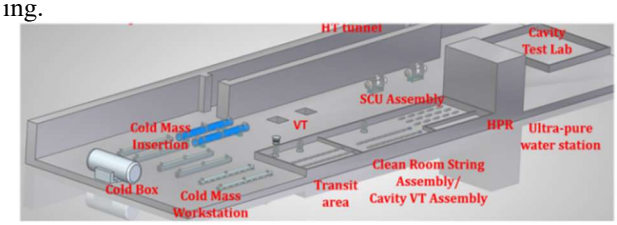


Figure 2: Layout of test facility.

The cryogenic component multi-functional test facility is used to test the working life of the tuner at cryogenic, the vacuum leakage rate and electrical performance at cryogenic of cold BPM, the working stability of superconducting magnet and current lead at cryogenic, the thermal load of each temperature zone, and the vacuum leakage rate of the coupler at cryogenic.

The superconducting cavity vertical test facility is used to test the performance of the superconducting cavity vertically to check whether the cavity has reached the design goal, so that it can meet the needs of engineering. The superconducting cavity horizontal test facility is used to test the performance of the cryomodule, check whether the cryomodule meets the design goal, and make it meet the needs of engineering. it includes cryomodule section and Feed Cap & End Cap.

CONTROL SYSTEM

The control system is responsible for the device control, data acquisition, functional safety, high level database or application, as well as network and computing platform. It will provide operators, engineers and physicists with a comprehensive and easy-to-use tool to control the components of cryomodule test facility.

* Work supported by Shanghai Municipal Science and Technology Major Project(Grant No. 2017SHZDZX02)
† chenguanghua@zjlab.org.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The control system will be mainly based on EPICS to reach the balance between the high performance and costs of maintenance. EPICS is a set of open source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as particle accelerators, telescopes and other large scientific experiments [3].

As shown in Fig. 3, the control system can be divided into there layers to ensure the performance and scalability, which are operator interface layer, input/output controller layer and device local control layer.

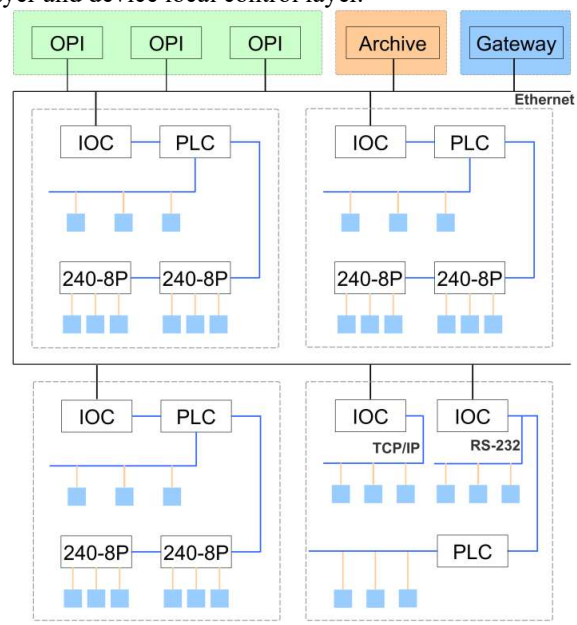


Figure 3: Architecture of the control system.

The local control layer is responsible for the monitor and control of temperature, pressure, cryogenic valve, liquid level, heater power supply, vacuum and so on. The signal of horizontal test facility is shown as Table 1. The multi-functional test facility and vertical test facility are similar.

Table 1: The Signal of Horizontal Test Facility

Signal	Qty	Type	
Temperature	120	Input	CX-1030-CU-HT
Pressure	2	Input	Analog
Liquid Level	3	Input	Analog
Cryogenic valve	4	Input	Analog
Cryogenic valve	4	Output	Analog, PID
Function safety	14	Input/Output	Digital
Heater Power Supply	16	Input/Output	Analog, Digital
Vacuum Gauge	3	Input	Serial port
Ion Pump	3	Input	Serial port
Magnet Power Supply	3	Input	Ethernet

TEMPERATURE MONITORING

Temperature monitoring(Fig. 4) is one of the key technologies for the test facility. LakeShore Cernox CX-1030-CU-HT is considered to be used as the temperature sensor(Fig. 5). Because the resistance of the cryogenic probe is not in the range of the PLC temperature monitoring module, temperature meter is used. The Cernox temperature probe is connected with temperature meter 240-8P by four wires to reduce error and reduce interference during transmission. 240-8P has 8 channels to connect temperature probe. The data from temperature meter is transmitted to PLC temperature module in sequence via profibus protocol. As shown in Fig. 6, F3LB01-0N is utilized as the model of PLC temperature module.

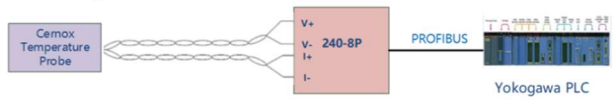


Figure 4: Temperature monitoring

Figure 5: Temperature sensor.

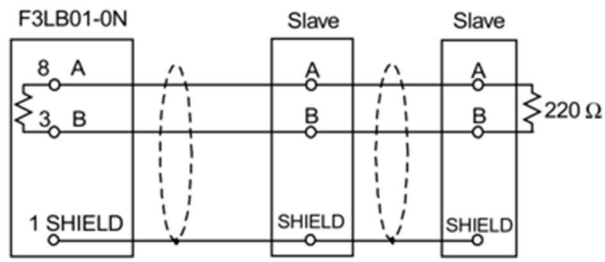


Figure 6: Structure of F3LB01-0N.

PRESSURE MONITORING

Pressure Transmitters(shown in Fig. 7) has 3 types applied to pressure monitoring. KELLER PAA-41X (0-100mbar, 4-20mA), KELLER PAA-33X (0-1600mbar, 4-20mA), and KELLER PAA-33X (0-3000mbar, 4-20mA). PLC module is Yokogawa F3AD08-4R (analog input module, 8 channels, 4-20mA), the internal circuit diagram of F3AD08-4R is displayed in Fig. 8.

Pressure Transmitters measures 2K helium circuit pressure(high pressure), 2K helium circuit pressure(low pressure), along with cryogenic pipeline pressure. 16-bit analog-to-digital module is selected, and the acquisition accuracy meets the equirement.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR04

Figure 7: Pressure transmitter.

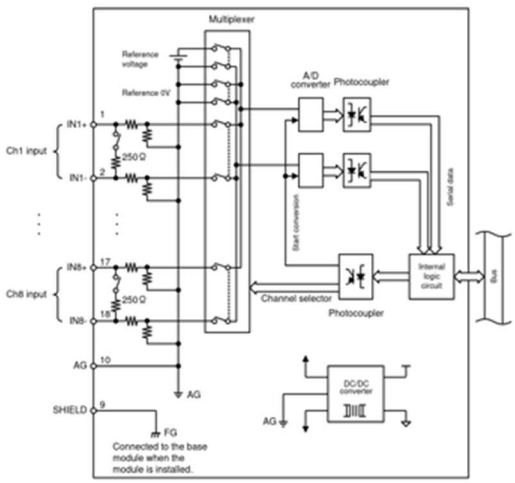


Figure 8: Internal circuit diagram of F3AD08-4R.

MONITORING AND CONTROL OF CRYOGENIC VALVE

The model of cryogenic valve selects Toko Valex T-8800, while the model of digital electropneumatic valve positioner(Fig. 9) picks SAMSON 3730-2. PLC modules are Yokogawa F3AD08-4R (analog input module, 8 channels, 4-20mA) and Yokogawa F3DA04-5R (analog output module, 4 channels, 4-20mA). The internal circuit diagram of F3DA04-6R is displayed in Fig. 10.

Cryogenic valve contains cooling valve and bypass valve. Its function is to control the flow of helium gas. When needed, open the valve to allow helium to enter the pipeline to cool the cryomodule.

Figure 9: Digital electropneumatic valve positioner.

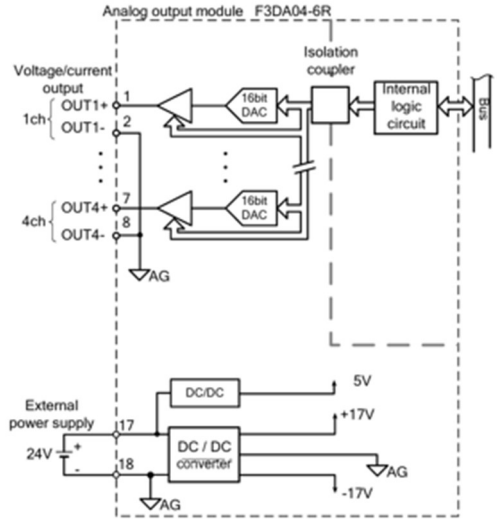


Figure 10: Internal circuit diagram of F3DA04-6R.

LIQUID LEVEL MONITORING AND STABILITY CONTROL

Liquid level monitoring and stability control(Fig. 11) is another key technology for the test facility. Level gauge has AMI 2K level probe. Liquid level gauge obtains AMI 1700-2K (0-10V analog output) as its model. The PID algorithm is used to realize the closed-loop control of the 2K liquid helium level. There are two control methods: manual control and PID control. The manual control is to set cryogenic valve opening on the interface while the PID control is that the PID program automatically sets the opening of the cryogenic valve. PLC module is Yokogawa F3CU04-1S (PID module, 0-10V analog input, 4-20mA analog output).

The final goal is to stabilize the height of the liquid level at a certain value. The liquid level gauge is partly in liquid helium and partly in air. The level gauge is connected with the liquid level acquisition instrument by four wires to reduce error. AMI 1700 is the liquid level acquisition instrument, it displays liquid level height, and it supports 4-20mA signal and 0-10V signal. AMI 1700 converts the height of the liquid level to voltage, and exports the voltage signal to the PID module F3CU04-1S. F3CU04-1S has its own algorithm, the parameters include setting value and output value. F3CU04-1S receives 0-10V voltage signal, which corresponds to the valve opening. It controls the valve dynamically according to the linear conversion relationship.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

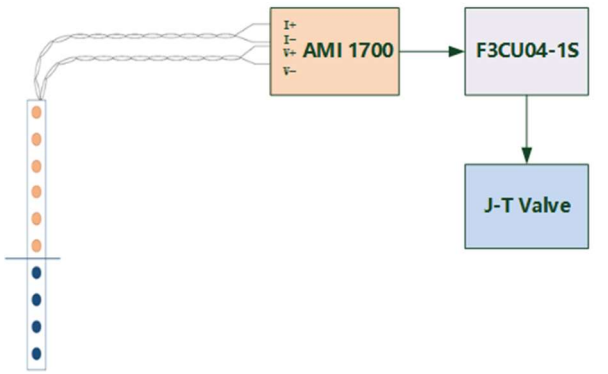


Figure 111: Liquid level monitoring and stability control.

As the algorithm diagram is shown in Fig. 12, e is the difference of setting value and target value. The target value is obtained from the liquid level sensor. u is the output of PID control, which conforms to the algorithm formula with 3 diagnostic parameters(k_P,T_I,T_D) in Fig. 13. These 3 parameters should be determined in the commissioning site.

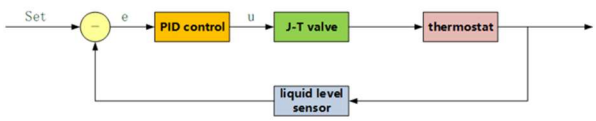


Figure 12: Algorithm diagram.

$$u(t) = K_P[e(t) + \frac{1}{T_I}\int_{s=0}^{t} e(s)\mathrm{d}s + T_D\frac{\mathrm{d}(e(t))}{\mathrm{d}t}]$$

Figure 13: Algorithm formula.

MONITORING AND CONTROL OF HEATER POWER SUPPLY

Omega KH-112/10 50W is utilized as the model of the heater, which is used for cryomodule heating up and cryomodule compensation of total thermal load power. Heater power supply model(shown in Fig. 14) is TDK Z60-3.5-C, the heater power supply has two modes (constant voltage and constant current), and constant current mode is used in this project. PLC includes Yokogawa F3XD16-3F DC digital input module, F3YC08-0C relay digital output module, F3AD08-5R 0-10V analog input module, and F3DA08-5R 0-10V analog output module. It realizes switching on/off power supply remotely, power failure status readback, and power operation mode readback (constant voltage, constant current). It achieves power supply output current value readback, power supply output voltage value readback, and power supply output power value readback. It fulfills current output setting. Constant control of the total thermal load power of the cryomodule has two methods: manual control and automatic control. The manual control is to set the current setting value on the interface while the automatic control automatically sets the current setting value, which is realized with Python + cothread. Cavity dynamic load + heater compensation output power = total power (constant).

Figure 14: Heater power supply model.

FUNCTION SAFETY CONTROL

The function safety logic is established between the system and the device. When a fault occurs, it can reliably and quickly protect the important components, send an alarm signal, and report the fault information to the control system. For input signals, the system can realize alarm latch, reset, and bypass processing from the software. Latch refers to saving the signal alarm state. Reset refers to releasing the alarm latch. Bypass means that the software temporarily releases the function safety response to the input signal. Yokogawa F3XD16-3F DC digital input module and F3YC08-0C relay digital output module are adopted for the PLC modules. The Interface of function safety control is shown in Fig. 15.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUBR04

Figure 15: Interface of function safety control.

IOC AND OPI

MOXA DA-682B running Linux/CentOS is chosen as IOC to communicate with local I/O devices with serial port, using Modbus/TCP protocol. We make use of PyDM (Python Display Manager) to build the graphic user interface. It is a new framework for building control system GUI based on PyQt, allowing us to make a rich desktop app including all types of buttons (push, toggle), images and a lot more. It is also convenient to add new functions to existing widgets and add new widgets. A user interface of vertical Dewar is shown in Fig. 16.

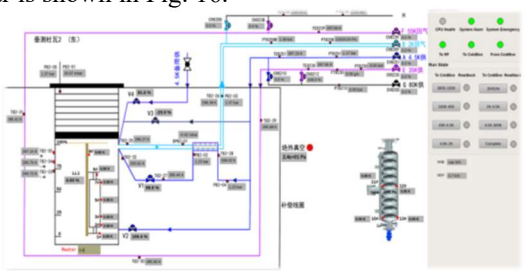


Figure 16: Control interface of vertical test platform.

Hundreds of EPICS PVs are archived by Archiver Appliance which is deployed on dual-server cluster as hot backup. Each server can provide data query service for all PV channels. Besides, with its excellent data retrieval performance, the archive client can be launched from a web page and used to browse and plot data as shown in Fig. 17.

Figure 17: Temperature of archiver appliance.

The control system of cryomodule test facilities has been operating stably for several months. The local control system makes use of Yokogawa PLCs and the corresponding algorithms, such as PID to keep liquid level and power balance. It is mainly used to monitor and control the parameters of temperature, pressure, liquid level, power of the heater and so on. The high level control system is based on EPICS to integrate all the devices in various systems, networked together to allow communication between them.

REFERENCES

[1] K. Li and H.X. Deng, "Systematic design and three-dimensional simulation of X-ray FEL oscillator for Shanghai coherent light facility", *Nuclear Instruments & Methods in Physics Research, A*, vol. 895, pp. 40–47, 2018, doi:10.1016/j.nima.2018.03.072

[2] SHINE project preliminary design report, 2017

[3] http://www.aps.anl.gov/epics/index.php

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV001

THE MIRROR SYSTEMS BENCHES KINEMATICS DEVELOPMENT FOR SIRIUS/LNLS*

G. N. Kontogiorgos†, A. Y. Horita, L. M. Santos, M. A. L. Moraes, L. F. Segalla,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Abstract

At Sirius, many of the optical elements such as mirror systems, monochromators, sample holders and detectors are attached to the ground with high stiffnesses to reduce disturbances at the beam during experiments [1]. Granite benches were developed [2] to couple the optical device to the floor and allow automatic movements, via commanded setpoints on EPICS [3] that runs an embedded kinematics, during base installation, alignment, commissioning and operation of the beamline. They are composed by stages and each application has its own geometry, a set number of Degrees-of-Freedom (DoF) and motors, all controlled by Omron Delta Tau Power Brick LV. In particular, the mirror system was the precursor motion control system for other benches [4 - 6]. Since the mechanical design aims on stiffness, the axes of mirror are not controlled directly, the actuators are along the granite bench. A geometric model was created to simplify the mirror operation, which turn the actuators motion transparent to the user and allow him to directly control the mirror axes.

INTRODUCTION

The latest Sirius mirror bench mechanical design version will be installed in Mogno and Ipê beamlines. They are composed by three stacked granites pieces (Fig. 1), the first one is supported by three levellers on the floor, the second is above the first and it has a ramp to form a wedge with the third granite.

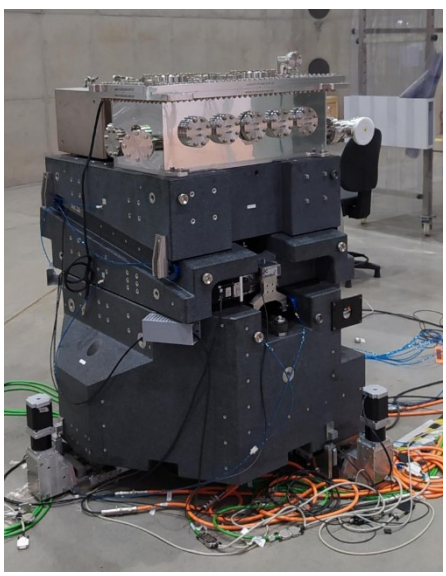


Figure 1: Granite assembly before installation and commissioning.

The mirror positioning depends on the relative motion between those components. Air-bearings were disposed both to guide and lift the granites between interfaces. The pressure, flow and activation of the air-bearings are commanded by a pneumatic panel.

KINEMATICS

Kinematic Model

As mentioned, the hole granite stack is supported by three levellers (The leveller is the component A of Fig. 2). They do not have feedback sensor, the position is measured by three Heidenhain gauges (Component B of Fig. 2) which touch the granites, so the measurement is done at the bench itself, not on the actuator. Although the controller can read the real position of the granite, this mechanical design allows the granite to slide over gauge and the leveller. This result presents incoherence between real position and the controller readback position. Furthermore, the geometry is much more complicated when considering friction and sliding on mechanics. To contour those problems, approximations were done on the geometry (As will be shown now) and an iterative actuation was developed (As will be shown later).

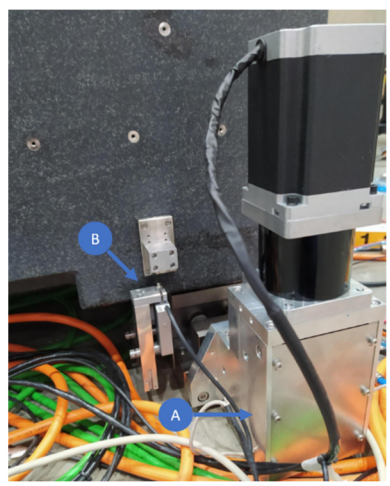


Figure 2: Leveller actuator (A) and feedback encoder (B).

The levellers could be interpreted as a parallel robot called tripod. The tripod was modelled using two reference frames [7], one rigidly coupled to the top platform (S_1), which is the moving one, and the other frame is at rest on the laboratory (S_0). Three vectors $\vec{r}_i(S_1)\ i = 1, 2, 3$, connect the S_1 origin to the three vertical Heidenhain sensors points that touch the platform. This is the first approximation of this model. Figure 3 illustrates both reference

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV001

frames, the vectors for each sensor and their projection on the sensors axes represented at laboratory reference frame.

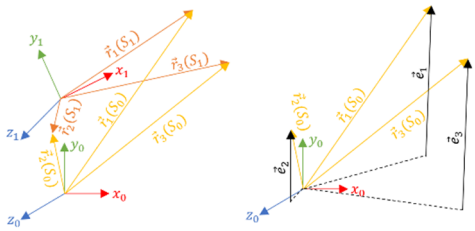


Figure 3: Referential transformation for tripod, part of mirror system kinematics.

The relationship between both reference frames is described by two rotations (R_x, R_z) and one translation about $\hat{y}_0$ axis (q_3), which are the allowed DoF for a tripod robot. Using the homogeneous transformation matrices T, it was possible to find the representation of the vectors rigidly coupled to S_1 on S_0 frame, the $\vec{r}_i(S_0)$ $i = 1, 2, 3$, which is the frame where the encoders are.

$$\begin{cases} \vec{r}_1(S_0) = T(R_x, R_z, q_3)\vec{r}_1(S_1) \\ \vec{r}_2(S_0) = T(R_x, R_z, q_3)\vec{r}_2(S_1) \\ \vec{r}_3(S_0) = T(R_x, R_z, q_3)\vec{r}_3(S_1) \end{cases} \quad (1)$$

The sensors movement are limited into $\hat{y}_0$ direction and the projection of the vectors represented on S_0 frame system onto $\hat{y}_0$ direction gives us the sensors positions $e_i, i = 1, 2, 3$.

$$\begin{cases} e_1(R_x, R_z, q_3) = |\vec{e}_1| = \hat{y}_0 \cdot \vec{r}_1(S_0) \\ e_2(R_x, R_z, q_3) = |\vec{e}_2| = \hat{y}_0 \cdot \vec{r}_2(S_0) \\ e_3(R_x, R_z, q_3) = |\vec{e}_3| = \hat{y}_0 \cdot \vec{r}_3(S_0) \end{cases} \quad (2)$$

Those relationships are usually called the inverse kinematics because they express the "actuator" position in function of the desired DoF's (on robotics these desired positions usually are the tool positions).

To couple the tripod to the granite actuators a transformation from parallel robot into serial robot was performed. The forward kinematics of the tripod is now necessary since each DoF represented in terms of the encoders could be transformed into an actuator of a serial robot. This set of equations are non-linear and finding the forward kinematics analytically is not possible. There are two ways of solving it: approximation by Taylor expansion and numerical methods. Since the angles are small and it would reduce the complexity of solution, the small angle approximation was chosen and the inversion of the linear system became possible.

The first air interface, shown on Fig. 4 item D, between the first and second granite has three actuators: one on the $\hat{x}_0$ direction (item C) and the others on the $\hat{z}_0$ direction (item B, the other cannot be seen from a single view because it is on the other side of the granite).

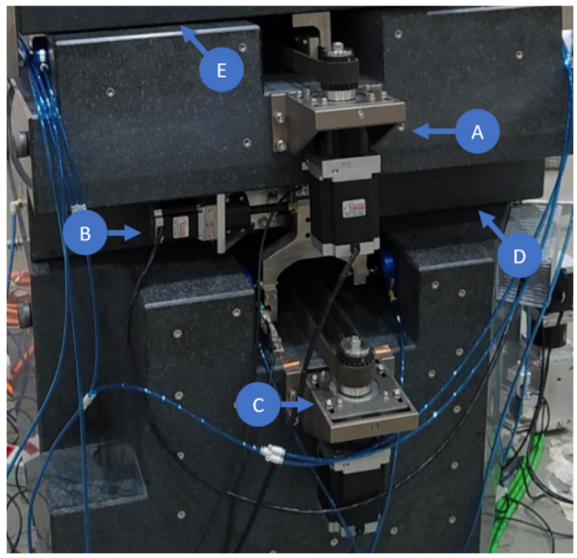


Figure 4: Granites' disposition of actuators and interfaces of granite bench.

Both $\hat{z}_0$ direction actuators can perform rotation (R_y) towards an axis parallel to $\hat{y}_0$ and translations (q_3) on an axis that is parallel to $\hat{z}_0$ if the motion is done in a coordinated way. The problem of transforming actuators into serial joints arises again since the z actuators forms a parallel robot. Some geometrical analysis led us to the following expressions:

$$\begin{cases} q_3 = \frac{z_{upper} d_{upper} + z_{lower} d_{lower}}{d_{upper} + d_{lower}} \\ R_y = \arctan \frac{\theta_{wedge}}{|\theta_{wedge}|} \frac{z_{lower} - z_{upper}}{d_{upper} + d_{lower}} \end{cases}. \quad (3)$$

Where z_i is the actuator position, d_i is the distance from the desired R_y pivot axis parallel to $\hat{y}_0$ axis to the $\hat{z}_0$ actuator and θ_{wedge} is the wedge angle (both $i = upper, lower$). This last parameter is only required to set the equations orientation, since the beamlines designers set the bench orientation according to their needs.

The last actuator (on granite bench because the internal mechanism has the piezo actuator) is the Fig. 4 item A, the wedge actuator, control the movement to lift the third granite through the ramp. Some caution must be taken here about the movement of the wedge, since it is used to lift the mirror parallel to $\hat{y}_0$ axis in the case that all rotations are zero. Although the pure movement may present undesired parasite motion, the hole inverse kinematics working together (this will be discussed further).

With all actuators properly modelled, a kinematics scheme could be synthetized in a serial robot model, Fig. 5.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

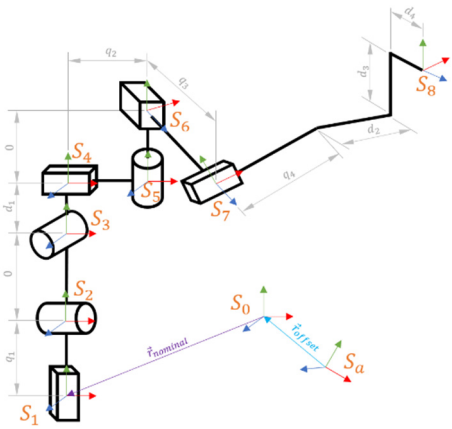


Figure 5: Mirror system kinematic scheme.

The scheme $\vec{r}_{nominal}$ is the fixed machine offset from the first joint to the control point S_8, usually set on the center of the mirror. This point is interesting because you are always commanding the positions to the mirror near the specified beamline mirror center. The $\vec{r}_{offset}$ is the user offset and was created to correct alignment positions or to set new positions to the mirror center as user want. The $q_i\,, i = 1 \ldots 4$ are the variable translations performed by the actuators and the $d_i\,, i = 1 \ldots 4$ are the fixed offsets between the joints. The levelers transformations results onto joints on S_1, S_2, S_3, the $\hat{x}_0$ direction actuator (Item C, Fig. 4) is the S_4 actuator, the composed movement in Eq. (3) with B actuators are performed by S_5 and S_6 joints and the last one, the S_7 joint is rotated by θ_{wedge} angle to model the wedge. The forward kinematics is totally modelled with this scheme and the rotations R_x, R_y, R_z and translations U_x, U_y, U_z could be found in function of the encoders.

Kinematics by Steps

There is no mechanism on levellers-granite interface that locks the parasite motion and the entire system can drift along undesired axes during some commanded movement. A hypothesis from mechanical design group is that a segmented movement instead of usual closed loop control would be a better approach to avoid this problem.

The proposal is setting the actuators in open loop while three encoders guarantee that each motor reach their specific position during the process.

The user would be able to define a threshold, which controls how close to the exact final position the levellers should get. At each iteration of the algorithm the position of the levellers is compared to the desired position, by reading the virtual motors. While their position is not inside the threshold band a new step is calculated based on the distance from the final position, them the motors are commanded to move several counts equivalent to the step. Once they reach this new position, the reference is compared again and the process continues. Figure 6 shows the state machine representation of the algorithm.

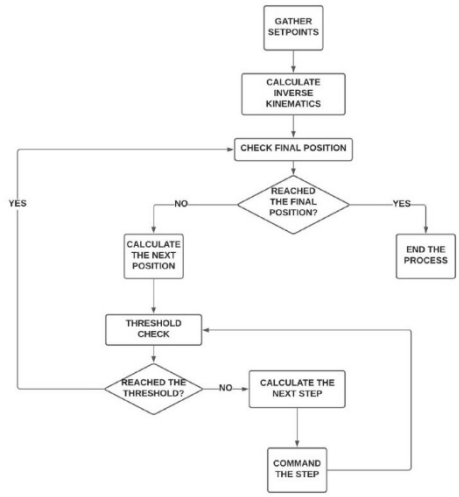


Figure 6: State machine representation of the kinematics by steps algorithm.

Numeric Solution for Inverse

The core of mirror kinematics is the inverse kinematics calculation since it gives the setpoints for actuators control loop. The user inputs for inverse kinematics are the DoF $\vec{u} = (R_x, R_y, R_z, U_x, U_y, U_z)$ of the granite bench at the control point S_8, related to S_a, the laboratory reference frame. The forward kinematics is a non-linear set of equations and finding its solution analytically is impossible. Then, the Newton method [8] was chosen for finding iteratively the encoders readings corresponding to some user input. The forward kinematics can be written as:

$$\vec{u} = \vec{f}(\vec{e}). \tag{4}$$

Where $\vec{e}$ is the actuators or encoders positions vector and note that $\vec{u}$ is the output vector on forward kinematics but it is the input on inverse as we are going to see. Let us define an error function $\vec{G}$, which is going to be minimized

$$\vec{G}(\vec{e}, \vec{u}) \triangleq \vec{f}(\vec{e}) - \vec{u}. \tag{5}$$

The method iteration can be defined as

$$\vec{e}_{k+1} = \vec{e}_k + \mathcal{J}^{-1}(\vec{e}_k)\vec{G}(\vec{e}_k, \vec{u}). \tag{6}$$

Where $\mathcal{J}$ is the Jacobian matrix of $\vec{f}$ and k is the iterator index. The iteration loop stops by the norm of $\vec{e}_{k+1}$ condition, which must be less than one count.

TESTS AND VALIDATIONS

The mirror kinematics validation was done by an external set of instruments. Six Heidenhain gauges were positioned in such a way to get the last granite position (Fig. 7).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV001

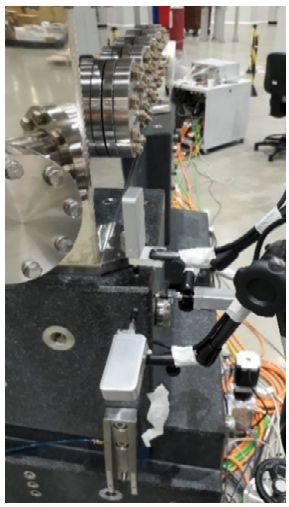


Figure 7: Setup for the kinematic validation.

Each granite surface which is touched by the gauges is modelled as a plane generated by three known points from the metrology data. Those points are fixed on the granite reference frame (e.g. this frame could be chosen) and a coordinate transformation with respect to $\vec{u}^{meas} = (R_x^{meas}, R_y^{meas}, R_z^{meas}, U_x^{meas}, U_y^{meas}, U_z^{meas})$ allows the representation of those points on the laboratory reference frame. The Heidenhain gauges measurement are represented by one of three coordinates of the plane equation while the other coordinates are the gauges' location on the plane, which has been obtained from metrology measurements.

The model described above gives the gauges positions in function of the coordinates $\vec{u}^{meas}$. The validation of kinematics can be done by the Newton-Raphson method, since $\vec{u}^{meas}$ can be found in function of gauges data. Each DoF was subject to a programmed movement on PowerPMAC using the developed kinematics and each gauge position was gathered with another PowerPMAC controller.

Some data were corrupted during the gather and not all analysis could be done in this paper, which is going to be done in future works. The useful data could be compiled in Table 1 and Fig. 8 illustrates the control data and post processed data from the measurement devices.

Table 1: Formatting of References

Parameter	Rx	Rz
Accuracy	3.44e-1	6.16e-1
Rx drift	3.44e-1	2.04e1
Ry drift	3.57e-2	5.71e-3
Rz drift	2.87e-1	1.79e1
Tx drift	2.94e3	1.01e1
Ty drift	1.95e1	1.70e2
Tz drift	8.64e1	1.52e2

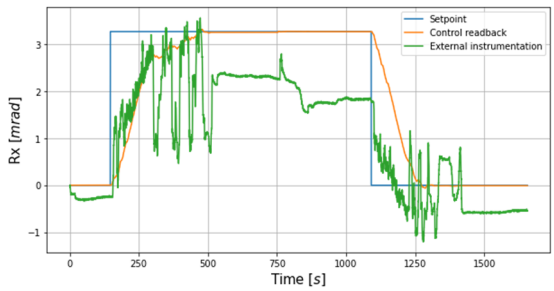


Figure 8: Measurement of a rotation towards x-axis done by mirror controller system and by validation method.

The accuracy is the difference of setpoint and the mean value of the measured data at the end of motion. The drifts are the difference from the position immediately before and the position after that this step motion is undone, which should be ideally zero.

CONCLUSION

The kinematics play an important role during beamline operation since it improves the user experience on its devices. Commissioning, alignment and operation are much easier when the user controls the mirror axes directly and no hand calculation is needed to find the positions of the actuators to perform a pure movement on the mirror.

Some tests were performed recently to characterize the motion of the system being operated by kinematics interface. A validation methodology was developed and now it can be improved and applied on the next beamline bases since some data was corrupted during tests before installation. Although there were problems, the present analysis offered several information for further project improvements.

REFERENCES

[1] C. S. N. Bueno *et al.*, "Vibration Assessment at the CARNAÚBA Beamline at Sirius/LNLS", *in Proc. MEDSI'20*, Chicago, USA, July 2021, paper MOPB08.

[2] R. R. Geraldes *et al.*, "The Design of Exactly-Constrained X-Ray Mirror Systems for Sirius", *in Proc. MEDSI'18*, Paris, June 2018, paper WEOAMA04. doi:10.18429/JACoW-MEDSI2018-WEOAMA04

[3] pmac, https://github.com/dls-controls/pmac.

[4] R. R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius", *in Proc. MEDSI'16*, Barcelona, Spain, Sep. 2016, paper TUCA05. doi:10.18429/JACoW-MEDSI2016-TUCA05

[5] L. Martins dos Santos *et al.*, "The Control System of the Four-Bounce Crystal Monochromators for Sirius/LNLS Beamlines", presented at ICALEPCS'21, Shangai, China, Oct. 2021, paper TUPV003.

[6] C. S. N. Bueno *et al.*, "Position Scanning Solutions at the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at ICALEPCS'21, Shangai, China, Oct. 2021, paper WEPV002.

[7] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York, NY, USA: John Wiley & Sons, Inc., 1989.

[8] M. A. Gomes Rugiero and V. L. Da Rocha Lopes, "Introdução à resolução de sistemas não lineares", in *Cálculo Numérico – Aspectos Teóricos e Computacionais*, Ed. São Paulo, SP, Brazil: Pearson Makron Books, 1997, pp. 197-200.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

MOTION CONTROL IMPROVEMENTS FOR THE KIRKPATRICK-BAEZ MIRROR SYSTEM FOR SIRIUS/LNLS EMA BEAMLINE*

G. N. Kontogiorgos†, C. S. B. N. Roque, M. A. L. Moraes, Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

The Kirkpatrick-Baez (KB) mirror system is composed of a vertical focusing mirror (VFM) and a horizontal focusing mirror. Both concave mirrors focus the X-ray beam by reflecting it at small grazing angles. The relocation of this system from UVX XDS beamline to Sirius EMA beamline facilitated a full revision of the motion control system, whose controller was migrated to Omron Delta Tau Power Brick LV. The beam focus is controlled by bending the mirrors through camshaft mechanisms coupled to low current Faulhaber motors. Although the amplifier is designed for higher currents, controller settings allowed the use of lower currents. Another improvement made is the ability to drive both bender motors in gantry mode and still control the lag between them. Each bender has a capacitive sensor to monitor the position of the center of the mirror, which is read by the analog input of the controller and made available by EPICS [1]. The VFM is supported by a tripod and a new kinematics was developed to reference the center of the mirror as the point of control. This paper presents the implementation of the new motion control KB system and its results at Sirius EMA beamline.

INTRODUCTION

The KB mirror system from XDS beamline at the previous accelerator UVX has been reconditioned for its operation at EMA beamline at Sirius. A full review of mechanisms and optics was done at the system since it was in operation for years at XDS. Some parts were adapted with new concepts brought from other Sirius devices, such as DCM and mirror systems [2,3]. The new parts give the system more repeatability since the backlash is reduced by the substitution of bushing-axis mechanism to parallel leaf spring translator. The mechanical enhancement favoured upgrades on the mirror bending control system.

A closed loop was developed using Power Brick controller for the mirror bending system. Since this controller has more features than the one used previously, the entire system, including its rough positioning by its supporting bench, has been redesigned. A kinematics inspired on mirror systems [4] was developed for a better user experience during commissioning, alignment, and operation because no hand calculation would be necessary to position the mirrors by commanding the actuators one by one.

The present paper is going to present further studies of driving low current stepper motors inspired on [5], the peripherical sensors that offer monitoring during operation and the details of kinematics implementation.

SYSTEM ARCHITECTURE

The KB system is composed of eleven motors with encoders, two piezos for fine pitch and two capacitive sensors. These are divided into two subsystems: VFM and HFM. Each subsystem is composed of a mirror with two motors responsible for driving the bending mechanism, with a capacitive sensor at the center of the mirror, and a base.

The HFM base is composed of one motor for each translation (Ux and Uy) and one for rotation around the X axis. The VFM base is more complex, composed of a tripod responsible for the required rotations (Rx and Ry) and translation on the Y axis.

All motors and encoders are controlled using a Power Brick LV. The capacitive sensors signals are read using the analog input in the same controller after being conditioned through the standard PI device. The piezos are controlled individually using PI servo controllers.

For the final users, all necessary commands are implemented in a graphic user interface (GUI) using process variables (PVs) made available by EPICS.

MIRROR BENDER

Driving Low Current Motors with High Current Amplifier

The Power Brick LV controller was adopted at Sirius for its ability to cover a range of motion control solutions. This controller is integrated with a 5 / 15 A amplifier [6]. The mirror bender motors used in the KB System are 250 mA and although the manufacturer does supply controllers with low current drivers, we opted to adapt the 5 / 15 A amplifier due to cost and time constrains. It has a bending system with two stepper motors AM2224 series model 0250 [7] coupled to a camshaft that pushes the mirror to bend and sets its focus.

Rising the inductance was the first setup to match the motor and controller specifications. To delay the current rising time, different values of inductors were placed in series with a low current test motor and the amplifier circuit could properly control the current on the motor coils.

Adding new components to the motor phases would change the motor properties, increase system cost, and decrease robustness. A detailed analysis led to the conclusion of limiting 48 V amplifier output PWM duty cycle. Since the current rise on the PWM on time and fall on the PWM off time [8], limiting the on-time voltage we chop the maximum reached current.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV002

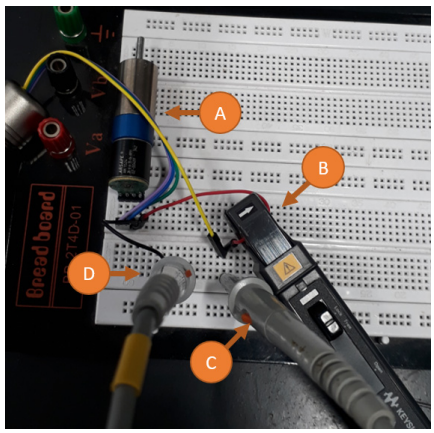


Figure 1: Proof of concept of current limiting by PWM scale factor.

Figure 1 shows the experimental setup for the validation of the duty cycle limiter concept. The phase A of ARSAPE AM1524 stepper motor A was submitted to a current bias of 100 mA from Power Brick. The current probe B was measuring the current passing on phase A coil and two voltage probes C and D were disposed to measure the voltage across the coil.

Starting from 95% of the full value of PwmSf, the percentual of the scale factor was reduced to zero and current measurements were performed at each scale factor value. The mean value of each measurement was calculated and plotted into the Fig. 2.

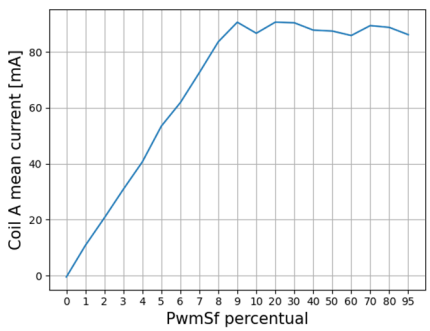


Figure 2: Percentual of full value of PwmSf versus Mean value of current at phase A submitted to a constant bias from the controller.

The test proved the current limitation, although the controller was trying to keep the current at 100 mA, it was not possible since PwmSf was limiting its output. This solution was chosen by its simple design. To ensure that the motor will not be damaged, phases fuses were displaced on final implementation.

Gantry Mode

To synchronize the bender motors movements, we created both the forward Eq. (1) and inverse (which is trivial from forward since the equation system is linear) native kinematic scripts implementing the following equations:

$$\begin{cases} X = 0.5(m_5 + m_6) \\ Y = 0.5(m_5 - m_6) \end{cases} \quad (1)$$

Where Y is the desynchronization angle and m_i are the motor positions. The X value is the bending parameter and can be interpreted as the bisector between the motor axes. Values set to the X parameter should move both motors in a synchronized manner, keeping the Y lag angle constant between them.

Capacitive Sensor Analog Input

There is a capacitive sensor behind each mirror to monitor the distance of the center of the mirror to infer the bending radius. An optical characterization of the mirror system was done using a Fizeau interferometer, which aims to find the relation between the mirror bending radius and the position read from the capacitive sensor.

This setup allows us to close the loop between bending motors and the capacitive sensor. Since the capacitive sensor does a direct measurement of the bending radius (in opposite to the bending motors encoders, which are coupled to the motor axis and perform an indirect measurement of the mirror bending radius) the control system would be more accurate on its setpoints. Closing this outer loop was not done yet due to the beamline deadlines but this is the next step on improving KB's positioning system.

VFM KINEMATICS

The VFM is supported by a tripod, a parallel robot with three degrees of freedom: Rx, Rz and Y, in Sirius reference frame. There is a linear stage coupled to the tripod that allows movements along the X axis. Those movements are required for the sake of simplicity on positioning the mirror on the correct place. The kinematics feature circumvents the need for iterative operation, improving time and user experience on the positioning of the mirror. Figure 3 shows the graphic user interface used with kinematics.

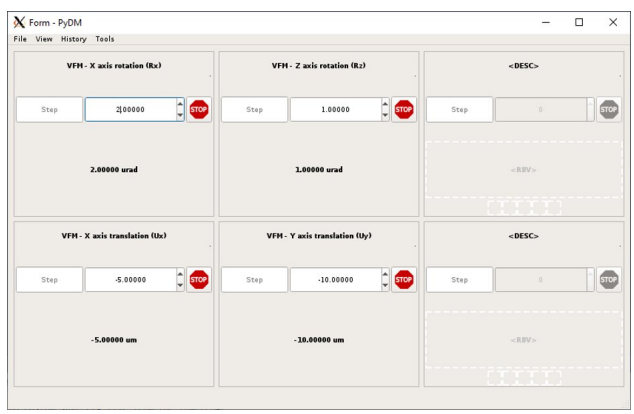


Figure 3: Test user interface for KB operation.

The stage above the tripod must be modelled together to obtain a single set of equations that describes the relationship between user positions and motor positions. This was done by transforming the parallel tripod robot into a serial robot with three joints, each one responsible for the three degree-of-freedom. Applying Taylor expansion on the tripod inverse kinematics (the angles are very small) turns the system easily inversible with a solver from the Python

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

SymPy library solver [9]. The operation resulted on the forward approximated equations and those positions could be translated to the robot joints. The full model was obtained by adding the linear stage into this serial version of tripod. Figure 4 shows the joint scheme derived from the system showed in Fig. 5.

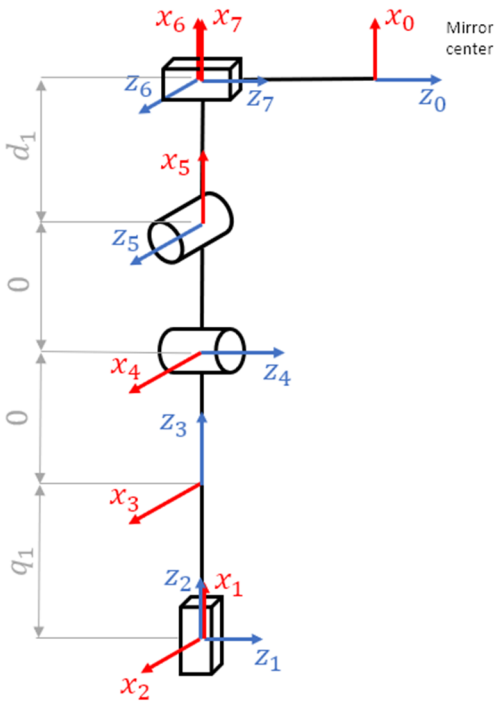


Figure 4: Kinematics joint scheme for geometric modelling.

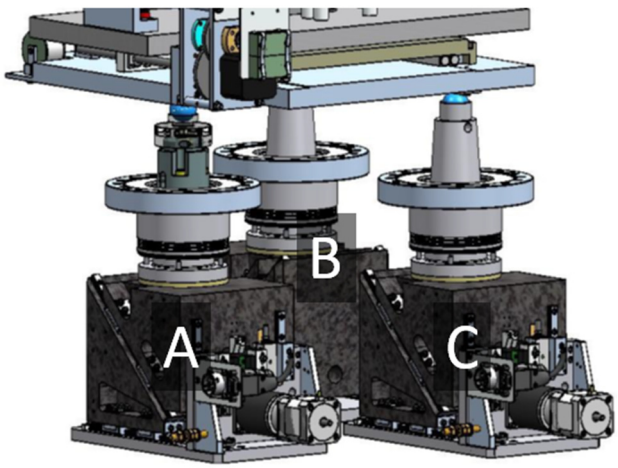


Figure 5: Tripod draw from system specification [10].

The KB solution uses the forward kinematics, which could be found by applying the basic transformation matrices on the robot joint scheme. The inverse kinematics is fundamental to command the motors with a setpoint input on Rx, Rz, Y and X axes and was found from the forward kinematics equations. The Power Brick embedded algorithm uses the Newton method [11] to solve the forward kinematics for the motors using as input the four user axes.

CONCLUSION

The EMA beamline KB system has been conditioned, it was successfully migrated to the new standard controller used at Sirius, the Omron Delta Tau Power Brick LV.

The studies and methodologies used on this work made possible and reliable to control low current motors with high current amplifier.

It was also possible to receive the capacitive sensors analog signals in the controller. This allows the simplification of the architecture and facilitates future improvements in closing the outer loop with this extra feedback.

The new kinematics used for the base of VFM improved usability and its equations were fixed, considering the reference frame transformation instead of simple geometric relations, as used before, which infer more accuracy to the motion.

To enhance the bending system and incorporate more accuracy on its movements, further work is going to set up a closed loop control using the capacitive sensor as feedback. This approach would measure the real bending radius instead of indirect measurements from the motors encoders.

At the time of this publication, the system has been commissioned and it is fully functional with the new improvements implemented.

REFERENCES

[1] https://github.com/dls-controls/pmac

[2] R. R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius", in *Proc. MEDSI'16,* Barcelona, Spain, Sep. 2016, pp. 141-146. doi:10.18429/JACoW-MEDSI2016-TUCA05

[3] R. R. Geraldes *et al.*, "The Design of Exactly-Constrained X-Ray Mirror Systems for Sirius", *in Proc. MEDSI'18*, Paris, France, Jun. 2018, pp. 173-178. doi:10.18429/JACoW-MEDSI2018-WEOAMA04

[4] G. N. Kontogiorgos *et al.*, "The Mirror System Benches Kinematics Development for Sirius/LNLS", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper TUPV001, this conference.

[5] O. Ivashkevych *et al.*, "Do You Really Need a Low Current Amplifier to Drive a Low Current Motor?", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUPHA134, pp. 730-733. doi:10.18429/JACoW-ICALEPCS2017-TUPHA134

[6] *Power Brick LV ARM User Manual*, Delta Tau Data Systems, Inc., Los Angeles, CA, USA, Dec. 2020, pp. 14-20, https://assets.omron.com/m/661730249d3863b4/original/Power-Brick-LV-ARM-User-Manual.pdf

[7] *Stepper Motors 22 mNm*, Faulhaber Minimotor SA, Croglio, Switzerland, Jan. 2021, https://www.faulhaber.com/fileadmin/Import/Media/EN_AM2224_FPS.pdf

[8] M. H. Rashid, *Power Electronics, Circuits, Devices and Applications*, Pensacola, FL, USA: Elsevier, 1993.

[9] https://www.sympy.org/pt/index.html

[10] *EMA KB Mirror Systems at CNPEM User Manual*, FMB Oxford, Oxford, Oxon, UK, July 2017, pp. 15-16.

[11] M. A. Gomes Rugiero and V. L. Da Rocha Lopes, "Introdução à resolução de sistemas não lineares", in *Cálculo Numérico – Aspectos Teóricos e Computacionais*, Ed. São Paulo, SP, Brazil: Pearson Makron Books, 1997, pp. 197-200.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV003

THE CONTROL SYSTEM OF THE FOUR-BOUNCE CRYSTAL MONOCHROMATORS FOR SIRIUS/LNLS BEAMLINES

L. Martins dos Santos*, J. H. Rezende, M. Saveri Silva, H. C. N. Tolentino,
L. M. Kofukuda, G. N. Kontogiorgos, P.D. Aranha, M. A. L. Moraes,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

CARNAÚBA (Coherent X-ray Nanoprobe) and CATERETÊ (Coherent and Time-Resolved Scattering) are the longest beamlines in Sirius – the 4th generation light source at the Brazilian Synchrotron Light Laboratory (LNLS). They comprise Four-Bounce Crystal Monochromators (4CM) for energy selection with strict stability and performance requirements. The motion control architecture implemented for this class of instruments was based on Omron Delta Tau Power Brick LV, controller with PWM amplifier. The 4CM was in-house designed and consists of two channel-cut silicon crystals whose angular position is given by two direct-drive actuators. A linear actuator mounted between the crystals moves a diagnostic device and a mask used to obstruct spurious diffractions and reflections. The system is assembled in an ultra-high vacuum (UHV) chamber onto a motorized granite bench that permits the alignment and the operation with pink-beam. This work details the motion control approach for axes coordination and depicts how the implemented methods led to the achievement of the desired stability, considering the impact of current control, in addition to benchmarking with manufacturer solution.

INTRODUCTION

The Four-Bounce Crystal Monochromators (4CM) was in-house designed to compose the set of optical systems in the longest beamlines in Sirius [1]: CARNAÚBA [2] (Coherent X-ray Nanoprobe) and CATERETÊ [3] (Coherent and Time-Resolved Scattering).

The energy is selected by two channel-cut silicon crystals whose angular position is given by two direct-drive actuators. A mask, mounted between the crystals, is used to obstruct spurious diffractions, having as actuator a linear stage.

The system is assembled in an ultra-high vacuum (UHV) chamber onto a motorized granite bench that permits the alignment and the operation with pink-beam.

The adopted motion controller for this system was the Omron Delta Tau Power Brick LV (PBLV) and his PWM amplifier. This works discuss the methods that led to achieve the requirements of stability and coordination, considering current control influence, and compares with manufacturer control solution, Aerotech Ensemble Epaq MR (Epaq).

* leandro.martins@lnls.br

SYSTEM ARCHITECTURE

Granite Bench

The granite bench is designed to both ensure high stiffness and allow the movement for alignment [4] of the monochromator and operation with pink beam, moving the UHV chamber to a position that the beam passes between the crystals.

Air-bearings in the bottom and top granite interface permits a frictionless motion in the translation in the X direction and rotation in the Y direction.

Furthermore, three levelers that supports the bottom granite compounds the translation in Y direction and rotations in X and Z directions.

The actuators of the granite bench are 2-phase stepper motors. The feedback of the position are made with a Heidenhain's quadrature incremental length-gauge for each leveler and a pair Renishaw's BiSS-C absolute linear encoder.

Figure 1 illustratrates the 4CM granite bench and his UHV chamber installed in CARNAÚBA beamline.

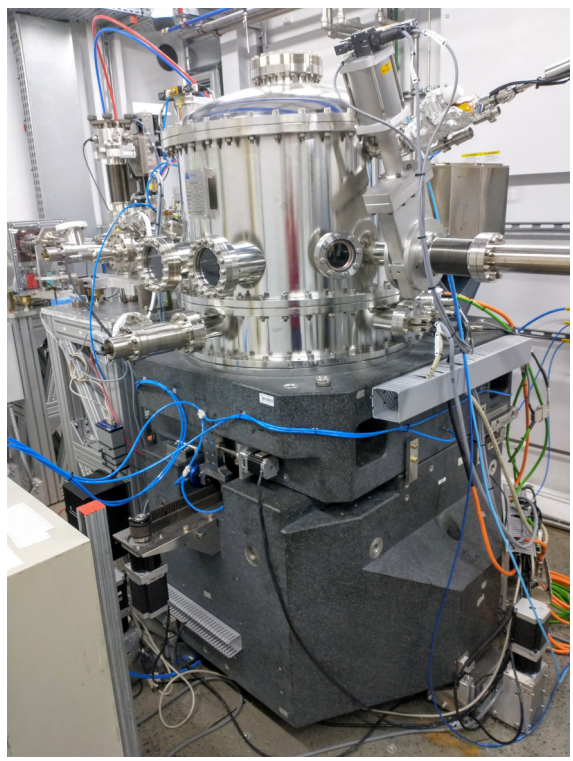


Figure 1: 4CM granite bench and UHV chamber.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV003

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Motion Controller

The motion controller used in this system is the PBLV. The PWM amplifier is able to provide currents of 5 A continuous and 15 A of peak and has maximum switching frequency of 30 kHz. It is able to decode differential quadrature and BiSS-C encoders, necessary in this project, besides others types of protocols [5].

It was considered the use of the Epaq, but for limitations encountered in the available features, the PBLV was chosen. As the standard motion controller for medium and high complexity applications in Sirius's beamlines, PBLV makes coordinating and triggering with other systems more compatible. In compensation, it was challenging to maintain the same Epaq performance, due to his linear amplifiers.

Crystals

The incoming pink beam is diffracted four times by two Si-111 channel-cut crystals, disposed in the configuration -++-. In this configuration, the crystals rotates in opposite directions, as seen in Fig. 2. In order to guarantee that the beam doesn't suffer any offset, is essential the channel-cut crystals have equal gap between his optical surfaces and the parallelism at each channel-cut.

From the Bragg's law, described in Eq. (1), we have the relation of the angle of diffraction (crystal angle θ) and wavelength (λ) of the output beam, being the first diffraction order and given the distance of crystallographic planes d = 3.135 416 1 Å for Si-111.

Equation (2), that describes the photon energy (E) from: Planck's constant (h), speed of light (c) and wavelength (λ).

Substituing Eq. (1) in Eq. (2), we get the energy (E) in function of diffraction angle (θ), as described in Eq. (3).

$$\lambda = 2d\sin(\theta) \tag{1}$$

$$E = \frac{hc}{\lambda} \tag{2}$$

$$E = f(\theta) = \frac{hc}{2d}\frac{1}{\sin(\theta)} \tag{3}$$

Each crystal are positioned by a direct-drive rotary stage, Aerotech APR200DR-155 (APR), indicated in Fig. 2 item d. It consists in permanent magnet synchronous motor (PMSM), Hall sensors and Renishaw's interpolated quadrature encoder system, with two readheads 180 ° spaced, each on with an angular resolution of 26.6 nrad [6].

Mask

A mask is used to prevent reflections and spurious diffractions contaminates the output beam. It must be positioned based on the beam offset caused by the first pair of diffractions, which is function of the angle of the first channel-cut angle (θ_1) and the gap between the optics surfaces, described in the Eq. (4) and illustrated in Fig. 2.

$$Offset = 2 \cdot gap \cdot \cos(\theta_1) \tag{4}$$

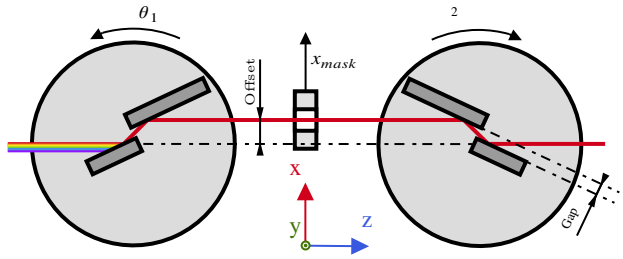


Figure 2: Representation of the crystals and mask. Indicating: the beam offset; the crystals gap; the adopted directions as positive for upstream crystal (θ_1), downstream crystal (θ_2) and mask; and the Sirius' coordinate system (x_{mask})

A screw-driven linear stage, Aerotech MPS50SL-025 (MPS) moves the mask between both crystal, indicated in Fig. 3 item c. In the configuration present at this application encloses a DC brush servomotor with 14:1 gearbox, a screw with 1 mm/rev of pitch, a rotary encoder with 512 lines/rev; resulting a resolution 34.8 nm at the load [7]. As the encoder is situated at motor's axis, one of the precautions taken during metrology procedures was to ensure that there is no considerable backlash . MPS also carries a diagnostic device, being a photodiode in CATERETÊ and a gold platted part in CARNAÚBA, that generates electrons by photoelectric effect.

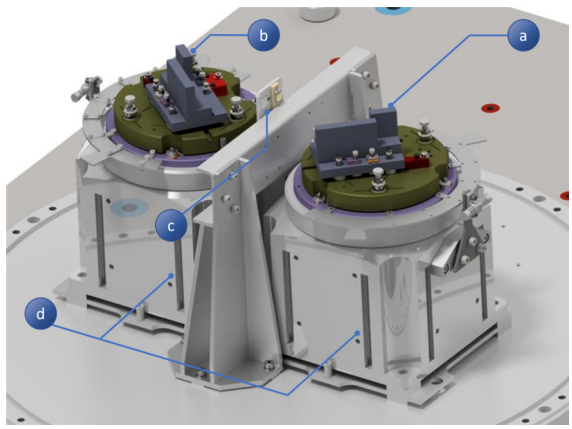


Figure 3: UHV components 3D model, namely: indicates upstream crystal; (b) indicates downstream crystal; (c) indicates the mask; (d) indicates the APR stages.

GRANITE BENCH KINEMATICS

The UHV chamber position is set by the relative motion of the stacked granite blocks performed by both motorized air-bearings and levelers mechanisms. The coordinated motion of all actuators enhances the user experience since the embedded kinematics [8] moves the control point in relation to the laboratory reference frame and no hand calculations of actuators position is needed.

The 4CM bench has a customized control point feature, playing an important role during alignment since the user

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV003

has three options for setting the reference frame[1]: rotation axis of the upstream or downstream crystal, and the midpoint between them.

DC BRUSH CONTROL

Having the DC brush a low rated current (0.16 A), was used the PBLV function PWM scale-factor, limiting the duty cycle [9] in 40 %, in order to avoid damage to the motor. A phase fuse was also included for redundancy.

The position control of the MPS stage have relatively more relaxed restrictions, due the usual small displacements and the size of the mask comparing to the beam size.

PMSM CONTROL

Phase Commutation

The phase commutation of the PMSM of the APR stage is made using the PBLV's algorithm, using the direct-quadrature-zero transform, common in field-oriented control (FOC).

As stage encoders are incremental, it is necessary to use Hall sensors for proper phase switching until the position is referenced by the home procedure; after this, having previously been characterized the rotor angle at position reference, the encoder is used as phase position feedback for higher resolution in commutation.

Current Control

Each phase of the APS stage have as electrical nominal specifications of resistance of 4.4 Ω and inductance of 1.7 mH. Low values of inductance motors are commonly susceptible to induced noise due current ripple caused by PWM drivers, like the one used in the PBLV.

In order to mitigate this kind of problem, a switching frequency of 20 kHz it was used. Even so, that did not prevent undesirable oscillations in the position reading.

Tests were made adding series inducers in each phase, but no significant improvement was observed, not even any correlation with the resultant inductance value.

The best results were achieved using the voltage-mode and adjusting the gains of the PBLV current-loop[2] to behave as a "pass-through" [10]. As trade-off, there is no proper torque control, what is not critical for this system.

Servo Control

The position and velocity control is made using the PBLV's standard servo-loop, which permits adjust digital filters, proportional-integrative-derivative (PID) and feed-forward gains.

Since the current-loop is not being closed for the APR stage, acting as a gain, the servo-loop was configured to be executed at the phase interruption, eliminating a full phase-cycle delay, what makes the servo-loop be executed at the same frequency of the phase algorithm, 20 kHz, improving even more the stability of the system.

On other hand, this configuration makes that encoder conversion tables are not processed, because they runs at the servo interruption, so the position and velocity feedback is processed at the same manner of the phase position [11]. The result of this is the loss of capacity of make average between the signals from both encoders present at the stage, a technique used to minimize errors from encoder interpolation and scale imperfections [12].

The stability of each APR stage was evaluating the position error, at a 20 kHz rate, for angles between 0° and 18° and the resulting cumulative amplitude spectrum (CAS) is presented in the Fig. 4. The values to each angle, integrating from 1 Hz to 2500 Hz, is as presented in the Table 1. Figure 4 make it clear that there are some well-defined frequencies that impact the stability of the system.

Table 1: Position Error CAS (1 Hz-2500 Hz)

Angle	Upstream Position Error	Downstream Position Error
0°	24.6 nrad	39.2 nrad
3°	24.6 nrad	33.3 nrad
6°	25.1 nrad	34.6 nrad
9°	20.2 nrad	34.6 nrad
12°	22.2 nrad	34.8 nrad
15°	21.6 nrad	39.1 nrad
18°	19.0 nrad	33.9 nrad

As a disclaimer, this test does not cover the whole stage position range, due to mechanical problems.

CRYSTALS COORDINATION

One of the most important requirements, besides the position stability, is the coordination between the crystals, in other words, how accurate is energy selected by the two channel-cut crystals. The tolerance is defined based in the crystal energy resolution and diffraction angle. Using the worst-case, the coordination error should be maintained below 1 µrad.

Epaq Coordination Modes

During the analysis of coordination using the Epaq controller, it was noticed that the suggested operation mode for this kind of operation, denominated electronic gearing, results in errors far beyond the acceptable limit for this application. This error occurs also in the difference of setpoints, indicating that there is a lag between the commands for both stages. The tests are at constant velocity of 1 °/s and results in coordination errors of approximately 210 µrad.

Another studied method was the gantry mode, in which satisfactory results were obtained around 1 µrad. However, this kind of operation sets one of the stages as the follower, disabling any input command from the user, that is necessary during the monochromator calibration. To toggle this mode,

[1] The control point could be set anywhere changing the kinematics parameters. Those predefined points were chosen by the beamlines' staff

[2] Integrative gain have no effect for voltage-mode

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV003

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

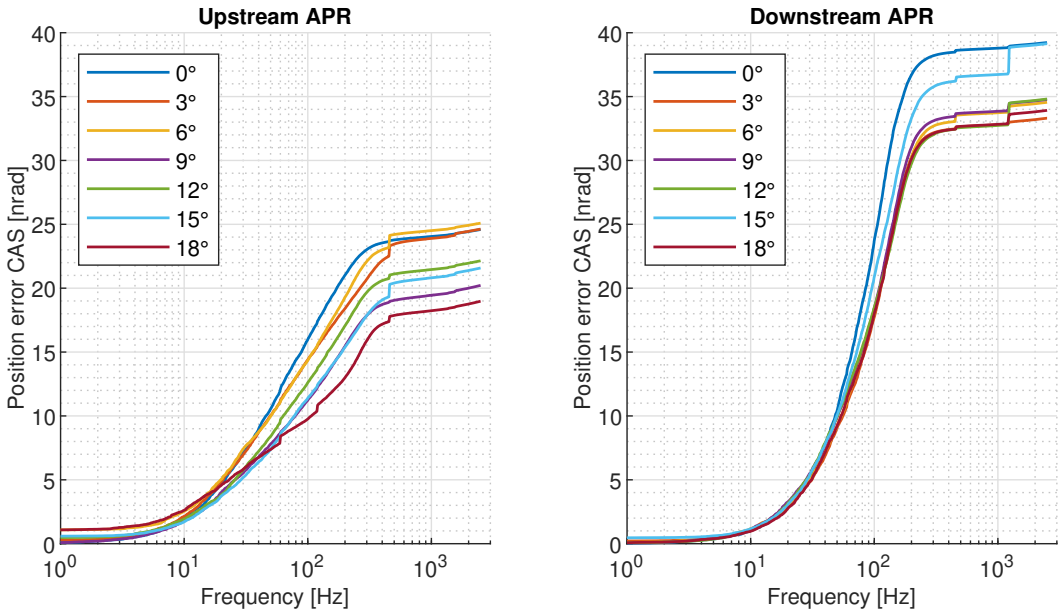


Figure 4: Cumulative amplitude spectrum in position error at each angle.

it is necessary to reset completely the Epaq, making loose the position reference. Also, with gantry mode enabled, the limits of the follower stage are not respected,being unwanted for system integrity.

Figure 5 compares this two modes, together with the coordination error when commanding both stage at the same command line, to the same position, namely uncoupled.

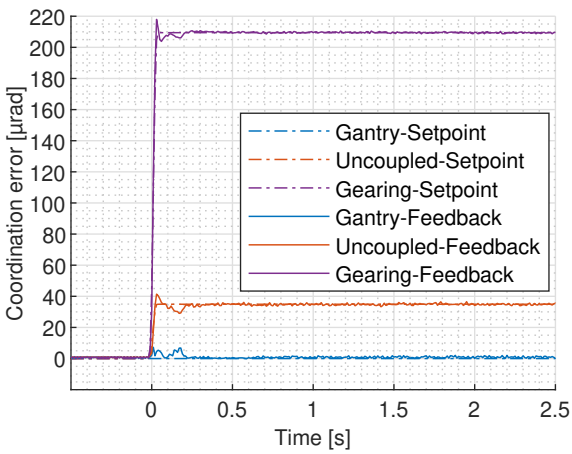


Figure 5: Ensemble Epaq MR coordination modes.

PBLV Coordination

In was used the native kinematics scripts for the transform energy (user input) in angle (in encoder counts units), also implements the coordinated movement between the crystals, being available an offset in the angle of each crystal for the monochromator energy calibration purposes.

Evaluating the coordination error at the same velocity of the tests with Epaq, 1 °/s, we can observe that, neglecting pike of 20 µrad occurred in acceleration period, the error is well behaved, remaining below 1 µrad, as seen in Fig. 6.

Possibly adding proper feedforward gains attenuate the error during the acceleration. Furthermore, it possible to use the PBLV feature of gantry cross-couple, causing the error of one of the stages to be taken into account in the servo-loop of the other.

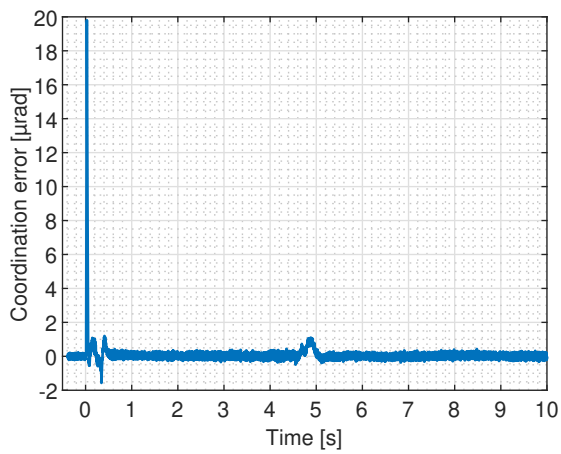


Figure 6: Power Brick LV coordination error.

CONCLUSION

The results of implementation of control for 4CM had satisfactory results, but there still scope for improvements-e.g. adjustmentment of feedforward and gantry cross-couple gains, coordination with undulator and investigation of noise sources.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV003

This work also put to the test the capabilities of the PBLV controller, which is of great value for new complex projects to be developed for Sirius.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, Innovation and the contribution of the LNLS team.

REFERENCES

[1] Sirius Project, https://www.lnls.cnpem.br/sirius-en/.

[2] Tolentino, H.C.N., et al., "X-ray microscopy developments at Sirius-LNLS: first commissioning experiments at the Carnauba beamline", in *Proc. SPIE 11839, X-Ray Nanoimaging: Instruments and Methods V*, 2021, p. 1183904.

[3] F. Meneau *et al.*, "Cateretê, the Coherent Scattering Beamline at Sirius, 4th Generation Brazilian Synchrotron Facility", in Coherence at ESRF-EBS, Grenoble, France, 2019.

[4] R. Geraldes *et al.*, "Granite benches for Sirius X-ray Optical Systems", in *Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation*, Paris, France, 2018. doi:10.18429/JACoWMEDSI2018-THPH12

[5] *Power Brick LV ARM User Manual*, Delta Tau Data Systems, Inc., Los Angeles, CA, USA, Dec. 2020, pp. 14-20, https://assets.omron.com/m/661730249d3863b4/original/Power-Brick-LV-ARM-User-Manual.pdf

[6] *APR Hardware Manual*, Aerotech Inc., PA, Pittsburgh, United States, Jun. 2018; https://www.aerotech.com/wp-content/uploads/2021/01/APR.pdf

[7] *MPS50SL Hardware Manual*, Aerotech Inc., PA, Pittsburgh, United States, Jan. 2020; https://www.aerotech.com/wp-content/uploads/2021/01/MPS50SL.pdf

[8] G. N. Kontogiorgos, A. Y. Horita, L. Martins dos Santos, M. A. L. Moraes, and L. F. Segalla, "The Mirror System Benches Kinematics Development for Sirius/LNLS", presented at ICALEPCS'21, Shangai, China, Oct. 2021, paper TUPV001, this conference.

[9] G. N. Kontogiorgos, C. S. B. N. Roque, and M. A. L. Moraes, "Motion Control Improvements for Kirkpatric-Baez Mirror System for Sirius/LNLS EMA Beamline", presented at ICALEPCS'21, Shangai, China, Oct. 2021, paper TUPV002, this conference

[10] *Power PMAC User's Manual*, Delta Tau Data Systems, Inc., Los Angeles, CA, USA, Jan. 2021, pp. 275-276; https://assets.omron.com/m/2c1a63d391d6bfa3/original/Power-PMAC-Users-Manual.pdf

[11] *Power PMAC User's Manual*, Delta Tau Data Systems, Inc., Los Angeles, CA, USA, Jan. 2021, pp. 302-303; https://assets.omron.com/m/2c1a63d391d6bfa3/original/Power-PMAC-Users-Manual.pdf

[12] Hayama, S., Duller, G., Sutter, J. P., Amboage, M., Boada, R., Freeman, A., Keenan, L., Nutter, B., Cahill, L., Leicester, P., Kemp, B., Rubies, N., and Diaz-Moreno, "The scanning four-bounce monochromator for beamline I20 at the Diamond Light Source", *Journal of synchrotron radiation*, vol. 25, no. 5, pp. 1556–1564. doi:10.1107/S1600577518008974

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

THE FPGA-BASED CONTROL ARCHITECTURE, EPICS INTERFACE AND ADVANCED OPERATIONAL MODES OF THE HIGH-DYNAMIC DOUBLE-CRYSTAL MONOCHROMATOR FOR SIRIUS/LNLS

R. R. Geraldes[1]*, J. L. Brito Neto, L. P. Carmo, E. P. Coelho, A. Y. Horita, S. A. L. Luiz, M. A. L. Moraes, Brazilian Synchrotron Light Laboratory (LNLS), CNPEM, Campinas, Brazil
[1]also at the CST Group, Eindhoven University of Technology (TUe), Eindhoven, The Netherlands

Abstract

The High-Dynamic Double-Crystal Monochromator (HD-DCM) has been developed since 2015 at Sirius/LNLS with an innovative high-bandwidth mechatronic architecture to reach the unprecedented target of 10 nrad RMS (1 Hz - 2.5 kHz) in crystals parallelism also during energy fly-scans. After the initial work in Speedgoat's xPC rapid prototyping platform, for beamline operation the instrument controller was deployed to NI's CompactRIO (cRIO), as a rugged platform combining FPGA and real-time capabilities. Customized libraries needed to be developed in LabVIEW and a heavily FPGA-based control architecture was required to finally reach a 20 kHz control loop rate. This work summarizes the final control architecture of the HD-DCM, highlighting the main hardware and software challenges; describes its integration with the EPICS control system and user interfaces; and discusses its integration with an undulator source.

INTRODUCTION

With performance numbers in the range of single nm and tens of nrad, the High-Dynamic Double-Crystal Monochromator (HD-DCM) is a high-end control-based beamline instrument for energy selection with fixed-exit monochromatic beam [1]. It has been developed at the Brazilian Synchrotron Light Laboratory (LNLS/CNPEM) for the 4th-generation Sirius light source [2] to be the first vertical-bounce DCM to reach, even in motion, 10 nrad RMS inter-crystal parallelism over the broad frequency range from 1 Hz to 2.5 kHz, representing improvements by factors between 3 and 100.

Due to its singular architecture, different aspects of the HD-DCM have already been detailed to the community: conceptual design, mechatronic principles and thermal management [3–5]; results of in-air validation of the core, together with system identification and control techniques in the prototyping hardware [6, 7]; offline performance of the full in-vacuum cryocooled system, including scans solutions [8]; dynamic modelling work, updated control design and FPGA implementation in the final NI's CompactRIO (cRIO) [9–11]; and calibration and commissioning procedures, together with the first experimental results with beam [12]. Here, updated schemes of the control system, as a result of operational maturity with two units at the MANACÁ and EMA undulator beamlines, and the emerging control-related strategies and bottlenecks concerning a holistic beamline operation are discussed.

* renan.geraldes@lnls.br

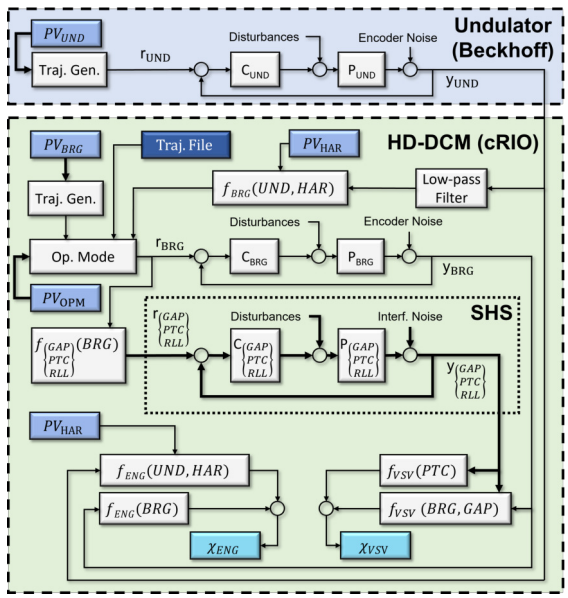


Figure 1: Simplified control diagram for the current HD-DCM integration at Sirius undulator beamlines.

APPLICATION OUTLINE

The application, as currently implemented at the EMA and MANACÁ beamlines, can be briefly introduced via the simplified control diagram in Fig. 1. Planar undulators by Kyma have been provided for the early operation phase, before the definitive Delta undulators that have been developed in-house for Sirius become available. Then, due to the perspective of short-time replacement, only basic features have been specified. Indeed, running on Beckhoff controllers, the systems can be operated by the users only via a limited set (PV_{UND}) of high level *process variables* (PVs) in the EPICS control system [13]. Complementary, the quadrature signal of the phase encoders have been derived to be used as the leading digital signals to the HD-DCMs, if desired.

The HD-DCMs can be defined by four main closed control loops running in NI's CompactRIO (cRIO) hardware. Firstly, BRG, with a bandwidth of 20 Hz, is responsible for controlling the angle of incidence of the beam on the so-called *first crystal*, for energy selection according to Bragg's law of diffraction. Then, GAP, PTC and RLL, with bandwidths between 150 and 250 Hz, are part of the so-called *crystal cage* (CCG), responsible for positioning the so-called *second crystal* with respect to the first one, so that fixed-exit monochromatic outgoing beam can be maintained at

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

different Bragg angles. The GAP controls the distance between the crystals, while PTC and RLL (for pitch and roll angles) control the inter-crystal parallelism. Due to a careful mechatronic architecture, in practice the originally *multiple-input-multiple-output* (MIMO) system can be statically decoupled, so that the loops may be addressed independently and simpler *single-input-single-output* (SISO) controller design techniques, such as loop-shaping, can be used (see [6, 10]).

The plants and controllers are represented as P_i and C_i, respectively, where i refers to UND, BRG, GAP, PTC and RLL. Each loop follows a reference r_i and provides a measurement y_i, being subject to actuator and plant disturbances, as well as measurement noise from sensors (encoders and laser interferometers). The most strict requirements for the HD-DCM are related to the errors ($e_i = r_i - y_i$) for: BRG, which ideally should be limited to the sub-µrad range to preserve the energy selection; and, more critically, PTC, specified as 10 nrad RMS (1 Hz-2.5 kHz) to preserve the beam position over long lever-arms at the beamlines (see [3]).

The reference signals for the CCG loops are defined as nominal or calibrated (linear or non-linear) functions f_i of the Bragg angle, being more suitably derived from r_{BRG}, rather than from y_{BRG}, to prevent disturbances and noise contamination which would deteriorate the CCG performances. The reference r_{BRG} itself, in turn, may result from three possibilities, namely: a trajectory calculated internally in the cRIO just before motion, a real-time conversion from a leading signal, or a trajectory previously stored in a file in the cRIO, thus defining three operational modes (PV_{OPM}).

The first is the *stand-alone* mode, in which the HD-DCM can be operated at a high level via EPICS PVs. With user-defined parameters, an internal block in cRIO generates a 3rd-order trajectory and r_{BRG} is updated accordingly to a target value. This is a basic mode for *stand-still* or *step-by-step scanning* experiments, either at bending magnet or undulator beamlines. In the latter case, as at EMA and MANACÁ, the undulator phase must be independently adjusted by the user, so that the appropriate tuning between its emission spectrum and the HD-DCM energy is assured for ideal flux and beam shape/size. Although not optimum in terms of timing, this can often be handled via standard step-scanning beamline pieces of software. Moreover, this would also be a straightforward option for the so-called *fly-scanning* continuous motion operation in bending magnet beamlines. Indeed, by reducing the previous state-of-the-art pitch parallelism levels at stand-still from about 50 to 10 nrad RMS, the HD-DCM can already provides more accurate beam delivery for a number of experiments, such as crystallography and imaging. Yet, it is by keeping these unprecedented high-performance numbers in fly-scans that scientific opportunities are created in spectroscopy, with typical experiment times reduced by up to one or two orders of magnitude for much higher throughput, and new strategies.

Next, in the *follower* mode, r_{BRG} is updated in real time as a nominal or calibrated function of a leading input signal, which can be either an external real-time trajectory generator or a direct instrument signal, such as the phase from an undulator source at an undulator beamline, which is the case for EMA and MANACÁ. Indeed, r_{BRG} can be completely determined by the undulator phase value, a desired harmonic number (PV_{HAR}) from the emission spectrum, and energy-conversion functions f_{BRG} . Then, by controlling the undulator alone, the user would ideally obtain a well-tuned fixed-exit monochromatic beam over the whole energy range, allowing the fly-scan potential to be explored as well.

Lastly, expanding flexibility in fly-scans and integration options, the HD-DCM can be potentially used in a *triggered* mode, in which arbitrary trajectories can be directly consumed from a file after a software or hardware trigger. At bending magnet beamlines, this would release boundaries related to the embedded 3rd-order trajectory generator in the HD-DCM, while preventing complementary real-time external trajectory generators for customized profiles. At undulator beamlines, in turn, not only may it also open opportunities with respect to possibly limited profiles in undulator controllers — which is the case with the Kyma commissioning undulators —, but, more importantly, it creates additional integration options, other than the control-loop-based ones that define the follower mode. The price to be paid is that more advanced following and synchronization functionalities must be provided by the undulator system as well, which has been considered for the Delta undulators.

Finally, complementary performance indicators can be computed in real time to indicate compliance. Firstly, the virtual source vertical shift χ_{VSV}, as a function f_{VSV} of the Bragg angle, the gap value and the pitch angle, is a useful figure, either in bending magnet or undulator beamlines[1]. Then, in undulator beamlines, such as EMA and MANACÁ, the mismatch χ_{ENG} in energy tuning between the source and the HD-DCM, as a function f_{ENG} of the Bragg angle, and the harmonic number and the phase, can be seen as a crucial parameter, particularly during fly-scans.

To conclude, as demonstrated in the *Results and Discussions*, it turns out that getting the measurement signal y_{UND} from the undulator currently brings significant practical limitations to the experiments, which, so far, have been non-ideally mitigated with an undesired aggressive low-pass filter for the encoder signal. All of this shines light on the increasing complexity and interdependence among the several instruments in new-generation beamlines. In particular, the results presented here will serve as guidelines for a more adequate architecture with the forthcoming Delta undulators and/or for upgrades in Kyma's. The main issues and alternatives are discussed in the following sections, after a brief review of the HD-DCM control hardware and software.

CONTROL ARCHITECTURE

During the prototyping phase, the HD-DCM mechatronic concept was validated in a rapid control prototyping tool in

[1] An indicator for the lateral shift of the virtual source χ_{VSH} could be computed as well, but, it has a more complex dependence on the Bragg and roll angles and, fortunately, is not a critically sensitive parameter.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

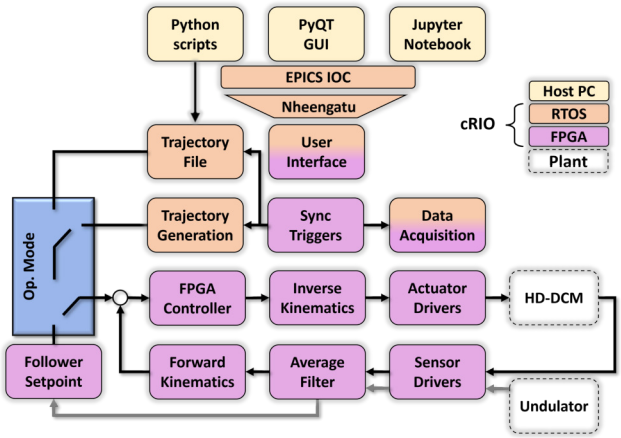


Figure 2: HD-DCM control architecture.

a real-time target machine from Speedgoat [7] – in which, besides preliminary operation, essential functionalities, such as plant identification and controller design via loop-shaping, were also developed [6]. To summarize the architecture, the FPGA layer was responsible for input and output drivers, while complex calculations were handled by its powerful processor at the real-time operating system (RTOS). However, in spite of being very flexible for rapid prototyping, this platform is not intended for reliable continuous operation, as demanded in a beamline environment. Therefore, NI cRIO was chosen as the standard controller for Sirius beamlines due to its industrial applications certifications and wide range of hardware modules, among other reasons.

The first architecture proposed for the control platform migration to cRIO was, naturally, the same one used in the Speedgoat xPC, but benchmark tests [11] revealed that the cRIO processor could not calculate the control loops sufficiently fast to meet the mechatronic requirements – namely, the high order controllers should run in a 20 kHz deterministic loop. Hence, an alternative approach with the whole control loop placed in the FPGA defined the final control architecture, which, as illustrated in Fig. 2, can be divided in three layers: FPGA, RTOS and Host Computer

Due to the results of the benchmark tests (see [11]), all the critical deterministic tasks are implemented in the FPGA layer. The *sensor drivers* are composed of decoders for BiSS-C absolute encoders, and incremental encoders and interferometers. Then, since the quadrature incremental signals demand high acquisition rates – specially when nanometric resolution is combined with relatively high velocity specifications –, the high frequency sensing noise is attenuated by decimation (or downsampling) from 10 MHz to 20 kHz in the *average filter*. Next, the *forward kinematics* converts individual sensor measurements into control coordinates, completing the feedback signal processing.

The reference signal depends on the selected operating mode. In the *stand-alone* mode, trajectories are first generated in the RTOS and internally executed by the *synchronization triggers* in the FPGA. Then, in the *follower* mode, the current integration architecture (see Fig. 1) is such that the *follower setpoint* calculates the BRG reference as polynomial functions of the averaged undulator encoder. Finally, in the *triggered* mode, trajectory files, previously generated via *Python scripts* and loaded in the cRIO memory (RTOS), are executed according to internal or external triggers that are managed by the *synchronization triggers* in the FPGA.

Once the reference and feedback signals are processed, the FPGA *controller* block can be executed. This implementation is particularly challenging because of the high-order controllers, broad numerical ranges, and high-resolution demands, especially in calculations. Hence, a new controller deployment workflow was created for standardization and reliability purposes, as described in next section. To complete the feedback control loop, the *inverse kinematics* distributes the forces and torques from control coordinates to actuator actions, which are passed by the *actuator drivers* to 3-phase brushless, stepper and voice-coil motors[2]. Complementary, two additional features implemented via FPGA FIFOs (first in, first out) are: a bidirectional *user interface* and a triggered *data acquisition* lossless logger for files.

Next, since all time-critical features are already addressed at the FPGA, no deterministic task relies on the RTOS layer, which is, then, used to manage four main tasks, namely: 1) the RTOS side of the *user interface*; 2) the *EPICS IOC* server implemented via *Nheengatu* [14] (further discussed in the following sections); 3) the generation or loading of *trajectories*; and 4) the RTOS side of the *data acquisition* file logger. Finally, working as an EPICS client, the Host Computer layer is responsible for providing user-friendly graphical interfaces (GUIs) and more sophisticated operational routines via Python scripts or Jupyter Notebooks, as discussed in more detail in the following sections.

Matlab-LabVIEW Integration Toolbox

The most challenging step for executing complex calculations in an FPGA is the trade-off between multiple hardware resources and timing constraints, like FPGA slices, memory blocks, lookup tables, Digital Signal Processors and timing violations, while taking care of the integrity of the results and proper numeric representations. That is the case in the HD-DCM controller, in which five SISO controllers[3], with up to 16 states each, process data at high loop rates and deliver forces and torques for a nanometric-precision motion control system.

Therefore, aiming at achieving a standardized procedure, a new workflow was defined, as depicted in Fig. 3. The first six steps are performed in Matlab and Speedgoat environments, whereas the last two using LabVIEW and cRIO: 1) *Plant identification*: stimulus and response analyses for system identification using *multisine* techniques. 2) *Loop-shaping controller design*: phase-margin and gain-margin analyses, with high low-frequency gain for disturbance attenuation and low high-frequency gain for noise rejection.

[2] The stepper motor is used in the long-stroke stage for the GAP, not discussed here for conciseness. For more details, see [3, 8].

[3] Again, the complementary long-stroke control loop was left out of Fig. 1 for the sake of clarity, but it, indeed, consumes resources in the hardware.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

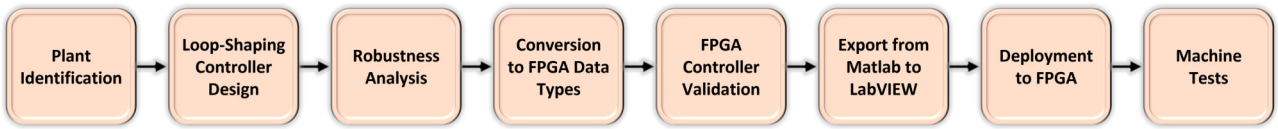


Figure 3: HD-DCM FPGA controller design and deployment workflow.

3) *Robustness analysis*: analyses of plots of open-loop Bode, Nyquist, sensitivity and characteristic loci for robustness and SISO validations. 4) *Conversion to FPGA data types*: reduction of high order controllers into lower order (≤ 4) transfer functions to preserve the transfer function integrity, with the required numeric representation reduction from double-precision floating point to single-precision floating point. 5) *FPGA controller validation*: testing of all transfer functions after data type conversion, to check if the increased quantization errors cause undesired changes in the dynamic system behaviour (Bode plot) (see [11]). 6) *Export from Matlab to LabVIEW*: file generation for exporting the transfer functions coefficients with no data loss. 7) *Deployment to FPGA*: updating of the multiple transfer function coefficients as constants into FPGA bitstream, as part of the strategy for saving hardware resources. 8) *Machine tests*: closed-loop controller validation using the real hardware, and performance analyses for compliance with specifications.

EPICS and Python

With the appropriate hardware, and low- and mid-level architectures validated in cRIO, an essential feature for the high-level user operation was having the HD-DCM control system compatible with EPICS, such that the day-to-day beamline work can be based on user-friendly graphical interfaces, or robust and efficient scripts.

With the standardization of the cRIO at Sirius beamlines, the Nheengatu solution has been internally developed by the Beamline Software Group (SOL) to integrate EPICS with cRIO (see [14]). In the HD-DCM, it is used for publishing cRIO variables as PVs in an EPICS IOC server running in the RTOS, as shown in Fig. 2. This occurs via two different communication interfaces, namely: inter-process communication for RTOS PVs; and NI's FPGA Interface C API for direct access to FPGA PVs. Most PVs are accessed through RTOS, except for the time-critical ones, like software triggers used for motion or data acquisition.

As a side note, it is worth mentioning that, although the architecture for time-critical PVs avoid delays related to real-time software processing, memory sharing between RTOS and FPGA, and also FPGA processing, it is important to keep in mind that the standard update rate in the EPICS server is only 10 Hz. So, hardware triggers are also essential for fast integration and synchronization at the beamline, having been implemented for low latency synchronization (update rates up to 9 MHz) for compatibility with the TATU FPGA-based solution that was also recently developed by the SOL group for Sirius beamlines [15] (see also [12]).

Next, the framework used to build the EPICS GUI is called PyDM (Python Display Manager) [16], which is based on PyQt and includes a variety of widgets (plugins) for QtDesigner for building GUIs that communicate with EPICS PVs. Besides these widgets, customized Python routines were also added to the GUI project for more specific behaviors. The main advantage of using PyDM is the possibility of rapidly assembling control system GUIs connected to EPICS PVs, and adding code when necessary in a natural, *pythonic* way. Also, PyDM has been widely used at Sirius for beamline operation GUIs, thus, knowledge, objects and routines could be reused for the HD-DCM GUI.

Then, to automate, standardize and speed up procedures and operations with the HD-DCM, several sets of Python scripts and Jupyter Notebooks have been developed. The structure is flexible, such that many tasks can be scripted independently, simply addressing EPICS PVs for different purposes. Yet, a more controlled Python library has also been created to handle generic tasks and sub-tasks, as well as essential routines and procedures. Among the most important library procedures are: homing, for machine start-up, which previously required trained human operation via GUI, but now can be executed robustly by any user via a script in only 3 minutes; and calibration routines with the X-ray beam, such as tuning with respect to the undulator and fixed-exit refinement (see [12]).

RESULTS AND DISCUSSIONS

For higher positioning accuracy and scanning throughput, the HD-DCM mechatronic architecture is based on what has been developed over the last decades for lithography machines in the semiconductors industry [17]. Now, to investigate the integration of the HD-DCM at the beamlines, examples in that field can also be explored as general references. Indeed, in [18], the leader-follower configuration for the reticle with respect to the wafer stage is conceptually equivalent for the HD-DCM with respect to the undulator in Fig. 1. However, fundamental differences can also be listed, including: type of motion; setpoint levels; and repeatability of tasks, motivating continued research.

One basic issue in adapting the strategy in [18] – i.e. with the measurement signal as input to the follower stage – to the HD-DCM is that there the wafer stage is also an ultra-low-noise nanometer-level system, whereas the undulator is an instrument that performs at the micrometer-level. Indeed, Kyma's encoder has a resolution of 0.5 µm, such that y_{UND} in Fig. 1 contains already a variation of a few microns peak-to-peak from quantization noise alone. When this is coupled to the HD-DCM, r_{BRG} becomes a signal with broad-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

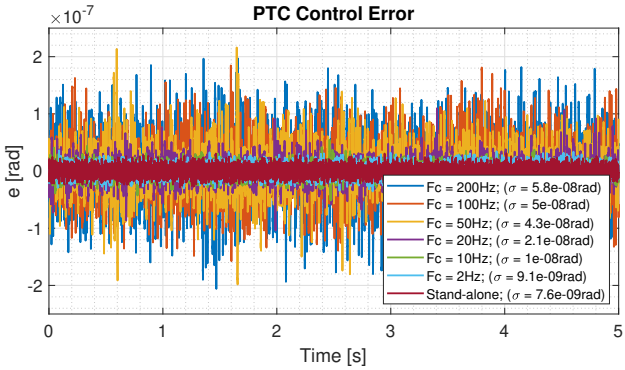


Figure 4: HD-DCM inter-crystal pitch (PTC) control error in stand-still condition at the EMA beamline at Sirius.

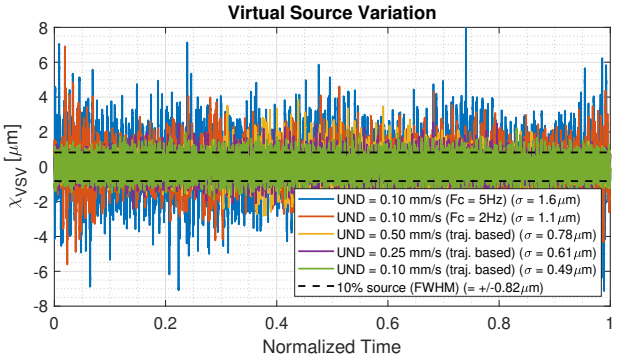


Figure 5: Virtual source vertical shift indicator χ_{VSV} in fly-scans of 1 keV at the EMA beamline at Sirius.

band noise reaching several µrad even at stand-still, critically deteriorating the HD-DCM closed-loop performance.

An example is given in Fig. 4, with measurements made at 6 keV at the EMA beamline, comparing the PTC control error in stand-alone and (low-pass filtered) follower modes. It can be seen that even with a Butterworth corner frequency as low as 2 Hz the RMS value is increased by about 20%, marginally respecting the desired spec of 10 nrad RMS. In this sense, some improvement can be expected by increasing the resolution of the encoder, thus reducing the noise background. Yet, this action alone would hardly completely solve the problem, because y_{UND} also captures the dynamics of the closed-loop undulator system, which may not have a sufficiently high performance due to its limited bandwidth and mechanical architecture.

Consequently, following y_{UND} during motion for fly-scans makes the problem even worse, as the disturbances in the undulator are increased. Another practical issue is that the low-pass filter adds delay to the reaction of the HD-DCM, detuning the emission spectrum and the Bragg angle, which may reduce the beam flux and/or distort its shape. A sensitivity analysis for tolerable margins for χ_{ENG} (not shown here due to space limitation) has been carried out via X-ray simulation tools by the Optics group, resulting in margins as low as ±4 µm for 10% increase in the beam full width at half maximum (FWHM). Within this budget, metrology accuracy, motion control errors, and delay effects must be considered. Hence, with typical motion errors measured within ±2 µm peak-to-peak (at least for velocities up to a few hundred µm/s) and the operation filter currently set 2 Hz, undulator velocities would be limited between 25 and 500 µm/s (depending on the undulator phase and harmonic) (if other errors are neglected). This translates to maximum scan speeds between 40 and 100 eV/s, which are at least 10 to 50 times slower than what can be expected under improved conditions, or in bending magnet beamlines.

This delay offset can be partly compensated via calibration, but practical limits are quickly found for more aggressive setpoints. Therefore, the ideal scenario probably involves directly extracting r_{UND} as the leading signal from the undulator controller, or making the undulator control system also capable of operating as a follower. Another alternative consists in linking the two systems solely via synchronized triggers, requiring that compatible trajectories be loaded in the different systems and executed accordingly. All these options are under investigation, but none of them are directly available with the current undulator solution. Still, in all cases, suitable velocity, acceleration and jerk parameters should be carefully identified to preserve performance.

An example of the fly-scanning capabilities is given in Fig. 5. The virtual source indicator χ_{VSV} is shown in follower mode for scans of 1 keV around 9 keV at the third harmonic of the Kyma undulator at the EMA beamline. Results for simulated smooth reference-based signals at different velocities and real encoder-based signals filtered at 2 and 5 Hz at the lower velocity are compared with 10% of the vertical beam size FWHM. The time axis has been normalized for readability, with the actual scans taking between 4 and 12 seconds. In the reference set, the peak-to-peak error increases by a factor 3 as larger speeds are associated with larger disturbances, yet, the standard deviation remains within to 10% of the beam size. In the encoder set, however, rougher setpoints and noise make the standard deviation exceed the margin even for the 2 Hz filter, which becomes worse with 5 Hz.

CONCLUSION

The efforts in designing and implementing the High-Dynamic DCM, as a high-end mechatronic machine at Sirius beamlines, have been rewarded with unmatching stand-still performance and scanning capabilities. The control framework relies on a complex multi-system architecture, which has been gradually optimized, from the FPGA drivers to the user experience and automatic calibration routines in Python/EPICS. Now, global integration bottlenecks start to be revealed, stimulating a more holistic approach towards beamline operation.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology and Innovation, and the contributions of the LNLS and the MI-Partners teams to this project. The HD-DCM is also a

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV004

PhD project carried out with the CST group at TUe, with Marteen Steinbuch, Hans Vermeulen and Gert Witvoet.

REFERENCES

[1] R. Geraldes, R. Caliari, M. Saveri Silva, T. Ruijl, and R. Schneider, *Instrument for Moving and Positioning of Optical Elements with Nanometric Mechanical Stability and Resolution in Synchroton Light Source Beamlines*, US Patent App. 16/331,925, Jul. 2019.

[2] *Sirius Project*, https://www.lnls.cnpem.br/sirius-en/, accessed: Oct. 1st, 2021, Oct. 30, 2021.

[3] R. R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius," in *Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'16), Barcelona, Spain, 11-16 September, 2016*, (Barcelona, Spain), ser. Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference, Geneva, Switzerland: JACoW, Jun. 2017, pp. 141–146, ISBN: 978-3-95450-188-5. DOI: 10.18429/JACoW-MEDSI2016-TUCA05. http://jacow.org/medsi2016/papers/tuca05.pdf

[4] R. R. Geraldes *et al.*, "Mechatronics Concepts for the New High-Dynamics DCM for Sirius," in *Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'16), Barcelona, Spain, 11-16 September, 2016*, (Barcelona, Spain), ser. Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference, Geneva, Switzerland: JACoW, Jun. 2017, pp. 44–47, ISBN: 978-3-95450-188-5. DOI: 10.18429/JACoW-MEDSI2016-MOPE19. http://jacow.org/medsi2016/papers/mope19.pdf

[5] M. Saveri Silva *et al.*, "Thermal Management and Crystal Clamping Concepts for the New High-Dynamics DCM for Sirius," in *Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'16), Barcelona, Spain, 11-16 September, 2016*, (Barcelona, Spain), ser. Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference, Geneva, Switzerland: JACoW, Jun. 2017, pp. 194–197, ISBN: 978-3-95450-188-5. DOI: 10.18429/JACoW-MEDSI2016-TUPE15. http://jacow.org/medsi2016/papers/tupe15.pdf

[6] R. M. Caliari *et al.*, "System Identification and Control for the Sirius High-Dynamic DCM," in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017*, (Barcelona, Spain), ser. International Conference on Accelerator and Large Experimental Control Systems, Geneva, Switzerland: JACoW, Jan. 2018, pp. 997–1002, ISBN: 978-3-95450-193-9. DOI: 10.18429/JACoW-ICALEPCS2017-TUSH203. http://jacow.org/icalepcs2017/papers/tush203.pdf

[7] G. B. Z. L. Moreno *et al.*, "Rapid Control Prototyping Tool for the Sirius High-Dynamic DCM Control System," in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017*, (Barcelona, Spain), ser. International Conference on Accelerator and Large Experimental Control Systems, Geneva, Switzerland: JACoW, Jan. 2018, pp. 1941–1946, ISBN: 978-3-95450-193-9. DOI: 10.18429/JACoW-ICALEPCS2017-THPHA214. http://jacow.org/icalepcs2017/papers/thpha214.pdf

[8] R. R. Geraldes *et al.*, "The Status of the New High-Dynamic DCM for Sirius," in *Proc. 10th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'18), Paris, France, 25-29 June 2018*, (Paris, France), ser. Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation, Geneva, Switzerland: JACoW Publishing, Dec. 2018, pp. 147–152, ISBN: 978-3-95450-207-3. DOI: 10.18429/JACoW-MEDSI2018-WEOAMA01. http://jacow.org/medsi2018/papers/weoama01.pdf

[9] R. R. Geraldes *et al.*, "Dynamic Error Bugdgeting in the Development of the High-Dynamic Double-Crystal Monochromator at Sirius Light Source," in *ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems*, (Boston, USA), ser. ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems, May 2020, pp. 119–124.

[10] R. M. Caliari *et al.*, "Loop-Shaping Controller Design in the Development of the High-Dynamic Double-Crystal Monochromator at Sirius Light Source," in *ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems*, (Boston, USA), ser. ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems, May 2020, pp. 125–130.

[11] M. A. L. Moraes *et al.*, "The FPGA Control Implementation of the High-Dynamic Double-Crystal Monochromator at Sirius Light Source," in *ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems*, (Boston, USA), ser. ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems, May 2020, pp. 131–136.

[12] R. R. Geraldes *et al.*, "Commissioning and Prospects of the High-Dynamic DCMs at Sirius/LNLS," in *Proc. 11th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'20), Paris, France, 25-29 June 2020*, (Chicago, USA), ser. Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation, Geneva, Switzerland: JACoW Publishing, Dec. 2021.

[13] *Experimental Physics and Industrial Control System*. https://epics-controls.org/, accessed: Oct. 1st, 2021.

[14] D. Alnajjar, G. Fedel, and J. Piton, "Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2020, WEMPL002. DOI: 10.18429/JACoW-ICALEPCS2019-WEMPL002.

[15] J. R. Piton *et al.*, "TATU: a flexible FPGA-based trigger and timer unit created on CompactRIO for the first SIRIUS beamlines," in *18th International Conference on Accelerator and Large Experimental Physics Control Systems*, this conference. DOI: THPV021.

[16] *Python Display Manager*, https://github.com/slaclab/pydm, accessed: Oct. 1st, 2021.

[17] H. Butler, "Position control in lithographic equipment [applications of control]," *IEEE Control Systems Magazine*, vol. 31, no. 5, pp. 28–47, 2011.

[18] S. Mishra, W. Yeh, and M. Tomizuka, "Iterative learning control design for synchronization of wafer and reticle stages," in *2008 American Control Conference*, IEEE, 2008, pp. 3908–3913.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

OPC-UA DATA ACQUISITION FOR THE C2MON FRAMEWORK

E. Stockinger*, B. Copy †, M. Bräger, B. Farnham, M. Ludwig, B. Schofield,
CERN Beams Department, 1211 Geneva, Switzerland

Abstract

The CERN Control and Monitoring Framework (C2MON) [1] is a monitoring platform developed at CERN and since 2016 made available under an LGPL3 open source license. It stands at the heart of the CERN Technical Infrastructure Monitoring (TIM) that supervises the correct functioning of CERN's technical and safety infrastructure. This diverse technological infrastructure requires a variety of industrial communication protocols. OPC UA [2], an open and platform-independent architecture, can be leveraged as an integration protocol for a large number of existing data sources, and represents a welcome alternative to proprietary protocols. With the increasing relevance of the open communication standard OPC UA in the world of industrial control, adding OPC UA data acquisition capabilities to C2MON provides an opportunity to accommodate modern and industry-standard compatible use cases.

This paper describes the design and development process of the C2MON OPC UA data acquisition module, the requirements it fulfills, as well as the opportunities for innovation it yields in the context of industrial controls at CERN.

INTRODUCTION

C2MON must consolidate information from a wide range of sources and so support a diverse set of devices and communication technologies through independent data acquisition modules. OPC UA is considered a key technology to managing heterogeneous machine interoperability and therefore fundamental to developments in advanced industrial environments [3]. As an open protocol, it addresses the common problem of insufficient interoperability by allowing equipment from different vendors to communicate with a host through the same interface [4]. Along with its high data modeling capabilities, OPC UA is a compelling candidate for adapting C2MON to modern use cases.

More generally, trends in industry urge Supervisory Control and Data Acquisition (SCADA) systems towards greater scalability and compatibility with Cloud Computing. While the C2MON server layer has already been adapted to natively support configurability injection and process monitoring [5], the OPC UA data acquisition (DAQ) module presented here pioneers these concepts on the data acquisition layer.

MONITORING OF THE DAQ PROCESS

The continuous monitoring of software plays a key role in ensuring smooth operation. Effective monitoring solutions enable a timely detection and treatment of issues related to quality-of-service and provide insight into system resource management [6]. The states or qualities to be monitored can be inherently complex and multidimensional, and so elude a direct measurement through single metrics. Kaner et al. [7] suggest the use of multidimensional metrics as a more meaningful representation of a complex quality.

C2MON provides extensive functionality for monitoring the elements within the supervised facilities. The connection between these elements and the processes on the C2MON data acquisition layer is also supervised through C2MON as a fundamental function. However, the DAQs are independent Java processes. Their internal states are not monitored through C2MON, but contain information relevant to ensure the availability, security and usability of the data acquisition infrastructure.

The metrics identified to be of relevance are diverse in nature. For example, the health of the machine and the operating system running the DAQ process can among others be tied to CPU, memory or disk usage, and the network load. Other metrics deal with the JVM and cover garbage collection threads and log messages. These are generic metrics which are provided by several external tools and client libraries. Other metrics are tied to the DAQ and the area of data acquisition. For example, these metrics could expose meta-information regarding OPC UA-specific concepts such as OPC UA subscriptions and their respective filter types.

Within the OPC UA DAQ process, metrics representing such internal states are exposed as observable endpoints through the Spring Actuator [8] project and Micrometer [9]. The metrics can be scraped out-of-the-box through the monitoring toolkit Prometheus [10] and be represented by a multi-dimensional data model. They can then be gathered and aggregated into a form enabling a global analysis by administrators and operators.

The choices of technological solutions were largely taken due to the prevalence of Spring within the C2MON infrastructure, and to be in line with the guiding principles of C2MON which prefer the use of proven technologies and of open-source resources where possible to facilitate reusability in other projects [1].

CONFIGURABILITY

There are two distinct aspects to configurability within the OPC UA DAQ. On the one hand, there is the need to configure the data to be monitored through C2MON. This use case is integral to the C2MON platform and impacts all tiers of the software architecture. This configuration data is stateful and managed on the application-level across the different DAQ modules and client applications. On the other hand, C2MON acts in heterogeneous environments with diverse hardware requirements. Operators and administrators

* elisabeth.stockinger@live.at
† brice.copy@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

benefit from a system catering to the high configurability of the OPC UA protocol.

Therefore the OPC UA DAQ module introduces a set of additional configuration options through the Java implementation of the Spring framework. Externalized Spring configurations can be specified in a variety of ways [11], profile specific application property files being the de-facto standard in the C2MON environment. This allows for the specification of different configuration profiles for example in between production and test environments, as well as an easy integration into container orchestration systems such as Docker [12] or Kubernetes [13].

The composition and configuration of required DAQ processes and the servers to be monitored is only known at runtime within the C2MON configuration strategy, and is susceptible to change. Therefore, handling configuration through Spring requires the dynamic creation and management of dedicated configuration scopes. The default Spring configuration scopes [14] do not support configuration options on the level of single connections to OPC UA servers out of the box. Therefore a custom Spring scope was defined to govern the life cycle of all software elements involved in the supervision of individual servers.

The OPC UA DAQ offers configuration options dealing among others with redundancy and connection settings, security, and modification strategies for information defined on the OPC UA servers.

In addition to monitoring, integration of Spring Actuators allows the remote management of applications through JMX and HTTP. Operators can for example change specific configuration settings on running processes, write data directly to OPC UA nodes or execute OPC UA nodes of the method type.

SECURITY

C2MON as a data communication network faces the risk of cyber-attacks and malware threatening confidentiality, integrity or availability of processes. One of OPC UA's distinguishing features among other low-level networking protocols is its extensive security model [15] which has been verified among others in [16] and [17].

OPC UA can be employed at different levels of the automation pyramid where the security requirements may differ. The uses of OPC UA as a communication interface can span from the isolated shop floor network to access via the internet as shown in Fig. 1. The different use cases are supported by OPC UA's layered approach to security, allowing a trade-off with other quality indicators such as performance or flexibility, depending on the needs of the environment.

OPC UA relies extensively on X.509 certificates [18] for security across these layers, addressing among others application authentication, user authentication and authorization, confidentiality, and integrity.

These certificates support a variety of trust models. Even within a single organization, several trust models can be in place, for example direct trust models where an administrator manages certificates on a case-by-case basis, hierarchical models, or fully-meshed Webs-of-Trust. Given that C2MON is a generic framework which must be adaptable to a variety of environments and use cases, the OPC UA DAQ module should support generic trust models and be configurable in its security modes. There are two major aspects to this solution: the client certificate must be configurable to comply with server requirements, and the server certificate must be processed and validated accordingly.

To satisfy generic use cases, the OPC UA DAQ module can be configured in one of three ways:

- the client certificate may be loaded from local storage,
- a self-signed certificate can be created on the fly, and
- connection can be established without a certificate.

Certificate validation is achieved through a PKI directory, a common approach in security management. This allows administrators to define their own solutions for certificate verification, be it through on-line or off-line methods.

While OPC UA offers rich potential in the area of user authentication and authorization, which would be valuable to exploit for any monitoring application context, this functionality is not currently supported within C2MON.

REDUNDANCY

Redundancy refers to the integration of several exchangeable instances in an environment. If one redundant component experiences faults, it can be replaced by another. Redundant setups can significantly increase component reliability and robustness and therefore are vital to critical systems.

OPC UA Part 4 [20] defines an extensive and versatile redundancy model encompassing redundant servers, clients, and networks, all of which can co-exist with one another. Server redundancy can be either *transparent* in which case fail-over cases between servers are handled exclusively on the server-side, or *non-transparent*, requiring the client to take action during a fail-over process. The server additionally has a *fail-over mode* which specifies the capabilities and behavior of the redundant setup, each mode imposing different requirements on the client. However, all redundancy modes defined within OPC UA can fall back to a lesser mode, where "Cold Redundancy" is the smallest common denominator.

Complex industrial environments often include servers which communicate through OPC UA but employ different and potentially proprietary models of redundancy. There are several such vendor-specific redundancy models within the context of TIM, demonstrating the need for standardization within industry.

While redundancy is increasingly relevant to ensure high availability, user demand for OPC UA redundancy is currently limited within the TIM environment. Most relevant to the artifact presented here is the use case of server redundancy, where the C2MON DAQ, acting as the OPC UA client, is presented with multiple sources of the same data.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

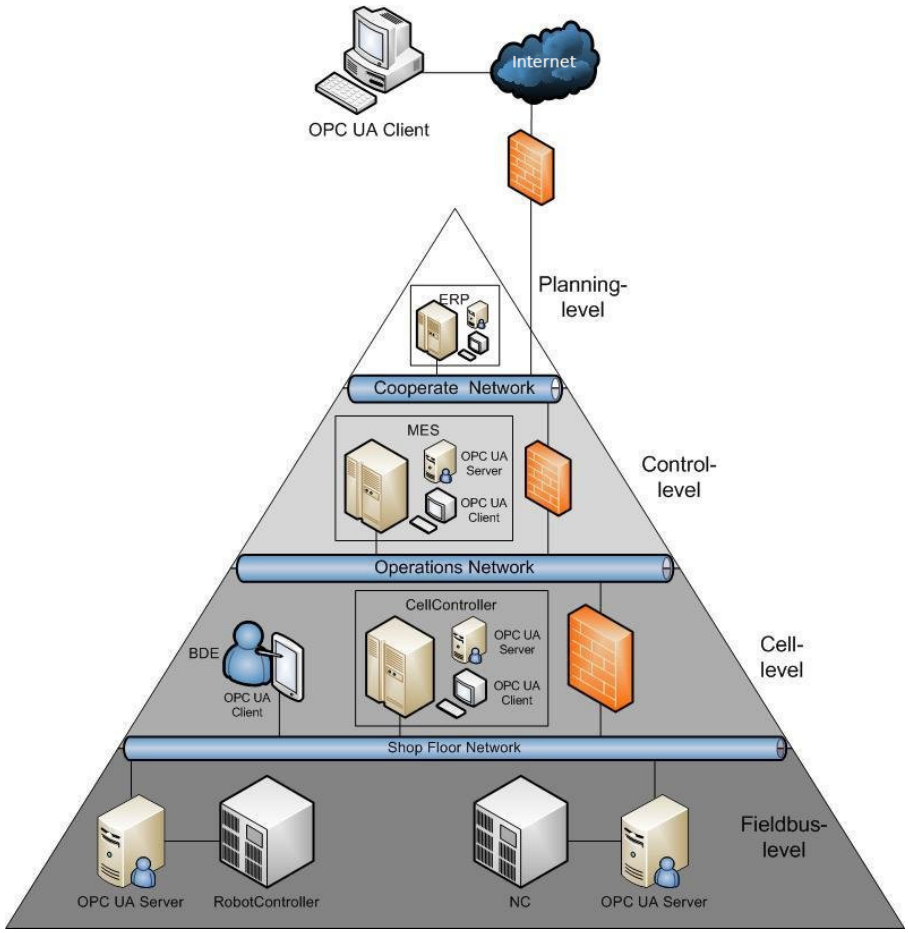


Figure 1: OPC UA as a communication interface in the automation pyramid, as depicted in [19].

For reasons of portability and testability, the design choice was made to only support redundancy in cold failover mode until there is user demand for other modes.

POTENTIAL AND PROSPECTS

The integration of OPC UA as a powerful open standard into C2MON has implications beyond a greater range of supported devices. There are several potential areas of future research and application within the context of CERN.

Configuration Management

C2MON configuration is passed inbetween the application layers at runtime via asynchronous messaging. This allows the dynamic addition and removal of entities to supervise. However, the configuration itself is static: configuration elements are mapped to specific DAQ processes, and there is only one configuration for each process. The static nature of the configuration of acquisition items is a potential obstacle in deploying C2MON within IoT projects: the conversion is Java-based, resource-intensive, and relies on proprietary services.

As an alternative to being passed between the architectural layers at runtime, it is possible to externalize the configuration. This approach enables a number of design patterns including service discovery, distributed configuration management, or load balancing. Similarly, configuration could be adapted to fit the capabilities and requirements of a DAQ when started within a dynamic environment. In the case of the OPC UA DAQ, instances could be configured to dynamically restrict subscriptions to data points producing very frequent updates on OPC UA servers which do not natively support deadbands. Similarly, data points with a low priority could be omitted if a server were to run low on memory during operation.

Spring supports the externalization of configuration natively. The Spring Cloud Config project [21] provides a dedicated platform for externalized configuration in a distributed system, which would integrate with a dynamic configuration handling within C2MON.

In order to benefit from Spring's support for externalized configuration within software components, these components must be managed by the Spring's inversion of control (IoC) [22], a mechanism abstracting the creation and management of component dependencies. The use cases described here are not currently supported by the C2MON framework core nor required within the TIM environment, but are only offered by the OPC UA DAQ module. The C2MON DAQ core initialization strategy historically relies on Java class

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

reflection, and predates the wide adoption of IoC design patterns in the C2MON technology stack. By externalizing the management of software components to Spring, we allow our OPC UA DAQ module to benefit from the rich options for configuration handling offered by the Spring ecosystem. This new approach to process configuration will hopefully be generalized to the C2MON DAQ core and benefit all other DAQ modules in the near future.

OPC UA as an Integration Platform within CERN

Equipment within a plant shop floor commonly offers several forms of communication, using proprietary communication protocols side-by-side with OPC UA or other open protocols. Redundant server setups may be supported natively only in configurations using proprietary communication protocols. Similarly, vendors may offer redundant server setups for OPC UA servers which rely on custom or proprietary architectural abstractions rather than the extensive redundancy model defined within the OPC UA standard. This non-standardized treatment negatively impacts system interoperability and integration.

Yielding to the prevalence of non-standard redundancy models, the OPC UA DAQ was designed to be extensible towards external models while supporting the OPC UA model of redundancy. The scope of extensions to vendor-proprietary models was restricted to initial connection, failover, and monitoring of connection status to ease the effort of switching from vendor-proprietary protocols to OPC UA without loss of functionality.

Likewise, only minor changes are necessary within the CERN environment to support OPC UA compliant certificates for increased security. CERN offers an established and well-accepted infrastructure regarding certificate authorities and certificate management, which is however not equipped to handle the creation of OPC UA-compliant certificates. Developers and administrators must therefore either forgo an integration of the CERN root certificate, or they must develop and maintain a custom add-on application polling the official revocation list and certificate authority, and handling certificate renewal.

The only factor preventing compatibility in the CERN certificate management services is the integration of certain extensions required by OPC UA. This issue would be easy to solve, thereby positively impacting the standardization of certificate creation and management across the organization.

OPC UA also offers functionality for user authentication and authorization. The exploitation of this functionality would open up new possibilities for auditing and access control. However, this would require C2MON to manage user authentication and authorization through a standardized interface across all DAQs and client applications. Such an interface is not available at present, among other reasons due to the stark differences in capabilities of supported protocols. The implementation of user authentication, authorization and of auditing are relatively open in the OPC UA specification, and out-of-the-box solutions offered by vendors are not cohesive. While this impedes the provision of standard auditing and access control functionality in heterogeneous environments, the integration of such an infrastructure is a worthwhile venture for future investigation.

Finally, OPC UA can reduce the complexity of an existing infrastructure whilst adding functionality. This is the case in some areas of alarm handling in the CERN infrastructure. Some systems supervised through the CERN C2MON instance declare alarms internally and publish them through JMS rather than publishing the data points themselves. Several intermediate steps are required to supervise these alarms in C2MON:

- The alarms must be transformed from into a JMS message that is compatible to C2MON.
- These JMS message are passed through a message broker to a process-specific DAQ.
- The filtered alarms are transformed into the C2MON "DataTag" format and passed to C2MON through the JMS queue.
- The alarms arrive at the C2MON cluster either in the state "on" or "off" and are further processed accordingly.

The devices exposing alarms in this fashion within the TIM environment offer communication through an OPC UA server. These data points can therefore be published directly to the new OPC UA DAQ module rather than as alarms to a separate and process-specific DAQ process.

This would allow the omission of the process-specific DAQ as well as the message format transformation, and so streamline the supervision pipeline in C2MON.

Alarms could in this fashion be defined and managed dynamically based on these data points using the rich configuration and validation functionality offered by C2MON client services [23, 24], avoiding consistency issues and increasing efficiency.

The current and proposed processes for handling alarms are contrasted in Fig. 2.

OPC UA as a Simulation Interface

The high complexity of services and processes acting within a SCADA system requires a holistic evaluation in a close-to-production setting to assure security and safety [25–28]. Ludwig *et al.* have found the sufficient validation of control systems for power supply at CERN not to be feasible using laboratory approaches [29]. They propose a simulation-based validation strategy for introducing new hardware into the environment using a custom simulation engine for power supply crates.

[29] describes a setup of redirecting traffic within OPC UA servers to target their so-called "VENUS" simulation engine instead of the physical crates, whilst keeping all other aspects of the SCADA systems in a production state. This allows for the validation of different hardware compositions within the production environment.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

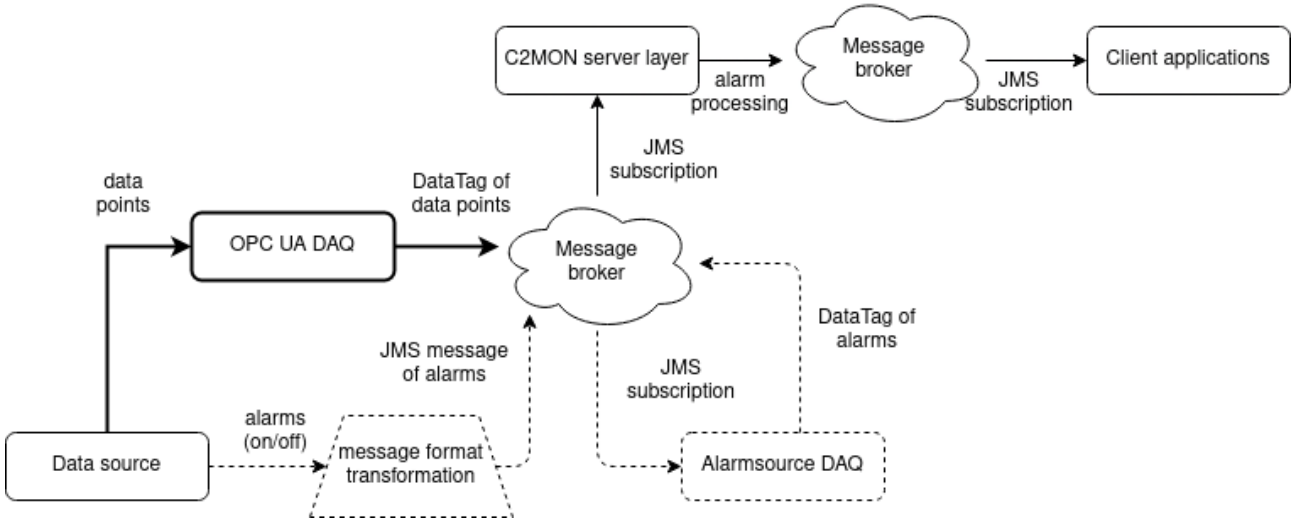


Figure 2: The pipeline of supervising alarms from systems currently publishing alarms as data points. Dotted elements indicate architectural blocks which could be omitted by exposing data points directly through OPC UA. Bold elements depict the alternative path through the OPC UA DAQ.

Rather than change the simulated hardware composition, it is also possible to trigger load and conditions exceeding those nominally observed in production. In this fashion, the SCADA layer can be examined for its tolerance to corner cases and exceptional circumstances. The OPC UA DAQ can so be used as an interface for validating the behavior of the SCADA system in adverse circumstances.

CONCLUSIONS

Along with extensive OPC UA integration, C2MON combines the powerful open-source Java management platform with OPC UA's rich data model and ubiquity in industry. The C2MON OPC UA DAQ module leverages OPC UA's potential as an integration platform on the plant level. Its support for Java process monitoring and its extensibility towards vendor-custom redundancy models makes it ideally suited to the ever-evolving demands of large-scale infrastructure monitoring. Finally, OPC UA can benefit a range of use cases within modern industry such as testing new hardware compositions within a simulated environment as well as testing the SCADA layer under simulated peak loads.

REFERENCES

[1] M. Bräger, M. Brightwell, E. Koufakis, R. Martini, and A. Suwalska, "High-Availability Monitoring and Big Data: Using Java Clustering and Caching Technologies to Meet Complex Monitoring Scenarios", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC140, pp. 439–442.

[2] OPC Foundation: OPC Unified Architecture: `https://opcfoundation.org/about/opc-technologies/opc-ua`

[3] I. González, A. Calderón. J. Figueiredo and J. Sousa, "A Literature Survey on Open Platform Communications (OPC) Applied to Advanced Industrial Environments", *Electronics*, vol. 8, no. 5, p. 510, Mar. 2019.

[4] J. Fitch and H.Y. Li, "Challenges of SCADA protocol replacement and use of open communication standards", in *Proc. of the 10th IET Int. Conf. on Developments in Power System Protection (DPSP'10)*, Manchester, UK, Mar. 2010, p. 35.

[5] B. Copy, M. Bräger, A. Papageorgiou Koufidis, E. Piselli, and I. Prieto Barreiro, "Integrating IoT Devices Into the CERN Control and Monitoring Platform", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA125.

[6] , A. Van Hoorn, J. Waller and W. Hasselbring, "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis", in *Proc. of the 3rd ACM/SPEC Int. Conf. on Performance Engineering (ICPE'12)*, Boston, MA, USA, Apr. 2012, pp. 247–248.

[7] c. Kaner and W.P. Bond, "Software Engineering Metrics: What Do They Measure and How Do We Know?", in *Proc. of the 10th IEEEInt. Software Metrics Symposium (METRICS'04)*, Chicago, IL, USA, Sept. 2004.

[8] Spring Boot Documentation on Actuators: `docs.spring.io/spring-boot/docs/current/reference/html/production-ready-features`

[9] Micrometer Application Monitoring: `micrometer.io`

[10] Prometheus: `prometheus.io`

[11] Spring Boot Documentation on Externalized Configuration: `docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-external-config`

[12] Docker: `www.docker.com`

[13] Kubernetes: `kubernetes.io`

[14] Spring Boot Documentation on Bean Scopes: `docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-factory-scopes`

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV005

[15] D.S. Pidikiti, R. Kalluri, R.K.S. Kumar and B.S. Bindhumadhava, "SCADA communication protocols: vulnerabilities, attacks and possible mitigations", *CSI Transactions on ICT*, vol. 1, no. 2, pp.135–141, Apr. 2013.

[16] OPC Foundation Security Bulletins: https://opcfoundation.org/security

[17] Kaspersky ICS CERT Security Analysis of OPC UA: https://ics-cert.kaspersky.com/reports/2018/05/10/opc-ua-security-analysis

[18] R. Housley, T. Polk, W.S. Ford and D. Solo, "Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.", *RFC Editor*, vol. 5280, pp.1–151, Apr. 2002.

[19] W. Mahnke, S.H. Leitner and M.Dann, "OPC Unified Architecture", Heidelberg, DE: Springer, 2009.

[20] OPC UA Specification: Part 4 - Services, Release 1.04: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-services

[21] Spring Cloud Config: spring.io/projects/spring-cloud-config

[22] Spring Documentation on the IoC container: https://docs.spring.io/spring-framework/docs/current/reference/html/core

[23] M. Bräger *et al.*, "Improving Alarm Handling for the TI Operators by Integrating Different Sources in One Alarm Management and Information System", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOPHA118, this conference.

[24] Z. Zaharieva and M. Buttner, "CERN Alarms Data Management: State & Improvements", in *Proc. ICALEPCS'13*, Grenoble, France, Oct. 2011, paper MOPKN011, pp. 110–113.

[25] R.L.Krutz, "Securing SCADA systems", Indianapolis, In, USA: Wiley Publishing Inc., 2005.

[26] M. Brändle and M. Naedele, "Security for process control systems: An overview", *IEEE Security & Privacy*, vol. 6, no. 6, pp.24–29, 2008.

[27] J. Slay and M. Miller, "Lessons learned from the maroochy water breach", in *Proc. of the 1st. Int. Conf. on Critical Infrastructure Protection (ICCIP'07)*, Hanover, NH, DE, Mar. 2007, pp. 73–82.

[28] M. Masera, I.N. Fovino and R. Leszczyna, "Security assessment of a turbo-gas power plant", in *Proc. of the 2nd. Int. Conf. on Critical Infrastructure Protection (ICCIP'08)*, Arlington, VA, USA, Mar. 2007, pp. 31–40.

[29] M. Ludwig, J.A.R. Arroyo Garcia, M. Bengulescu, B. Farnham, P.G.J. Gonzalez Jimenez, and F. Varela, "Developing and Validating OPC-UA Based Industrial Controls for Power Supplies at CERN", in *Proc. 12th International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'18)*, Hsinchu City, Taiwan, Oct. 2018, pp. 35–37. doi:10.18429/JACoW-PCaPAC2018-WEP04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV006

CONTROL SYSTEM OF THE SPIRAL2 SUPERCONDUCTING LINAC CRYOGENIC SYSTEM

A.Trudel, Q.Tura, G. Duteil, A. Ghribi
Grand Accélérateur Nat. d'Ions Lourds (GANIL), Caen, France
P. Bonnay
Commissariat à l'Energie Atomique (CEA/INAC) INAC, Grenoble, France

Abstract

The SPIRAL2 cryogenic system has been designed to cool down and maintain stable operation conditions of the 26 LINAC superconducting resonating cavities at a temperature of 4.5 K or lower. The control system of the cryogenic system of the LINAC is based on an architecture of 20 PLCs. Through an independent network, it drives the instrumentation, the cryogenic equipment, the 26 brushless motors of the frequency tuning system, interfaces the Epics Control System, and communicates process information to the Low Level Radio Frequency, vacuum, and magnet systems. Its functions are to ensure the safety of the cryogenic system, to efficiently control the cooldown of the 19 cryomodules, to enslave the frequency tuning system for the RF operation, and to monitor and analyze the data from the process. A model based Linear Quadratic regulation controls simultaneously both phase separators the liquid helium level and pressure. This control system also makes it possible to perform a number of virtual verification tests via a simulator and a dedicated PLC used to develop advanced model based control, such as a real time heat load estimator based on a Luenberger Filter.

INTRODUCTION

Cryogenic System

Spiral2 accelerator delivers high intensity beams of various ions for research in nuclear fields. It is mainly composed of a LINAC (LINear Accelerator) composed of 26 accelerating cavities installed in 19 cryomodules, made of bulk niobium and immersed in a liquid helium bath.

The cryogenic system, as shown in Fig. 1, is split in two levels: at the ground level stands the cryoplant with its refrigerator and helium collecting system, and in the underground accelerator tunnel, the 19 valves boxes form the cryogenic lines and feed the cryomodules with liquid helium [1].

Type-A and Type-B Cryomodules

The type-A and type-B cryomodules respectively contains one and two accelerating cavities. The helium bath is controlled through three valves shown in Fig. 2. The valve CV001 is used to fill the cryomodule by the bottom during the cooldown whereas CV002 and CV005 ensure respectively level and pressure regulation in the bath, which are measured by the LT200 and PT001 sensors. CV010 serves the shield outlet temperature regulation around 60 K.

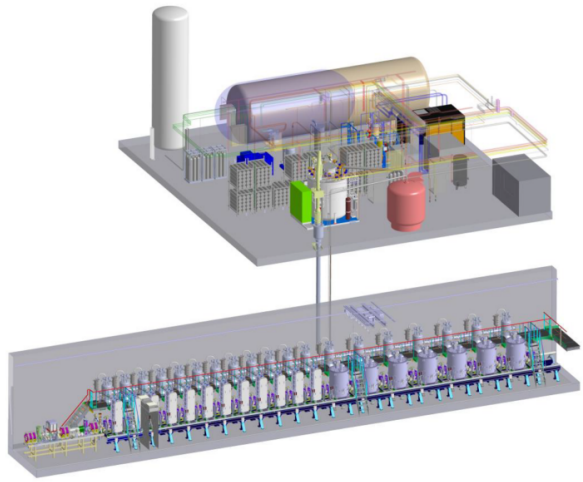


Figure 1: View of the Spiral2 cryogenic system.

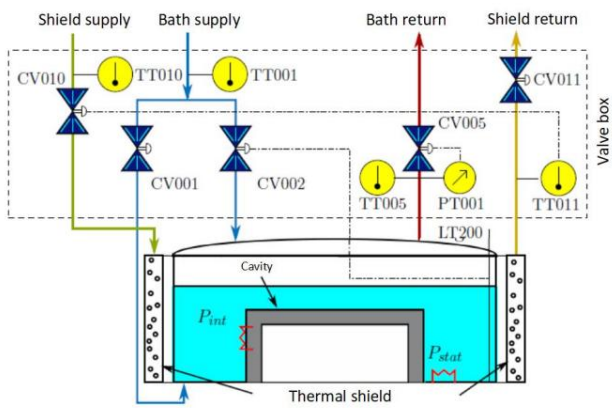


Figure 2: Type-A cryomodule and its associated valves and sensors.

ARCHITECTURE OF THE CONTROL SYSTEM

Liquid helium production and routing, as well as the automated safeties and the controls of the helium liquid bath, are ensured by a PLC-based control system. Data are issued from those PLCs to the Spiral2 Epics system for storage.

The core of the Linac cryogenics control system illustrated in Fig. 3 is made up of a fleet of 20 Siemens PLCs, one for each cryomodule and another one serving as a hub to ensure communication between the cryomodules and the outsides systems, three WinCC Pro supervisions, 1 workshop terminal, 26 brushless motors driving the Frequency Tuning Systems (FTS) of the RF cavities. In addition, there

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV006

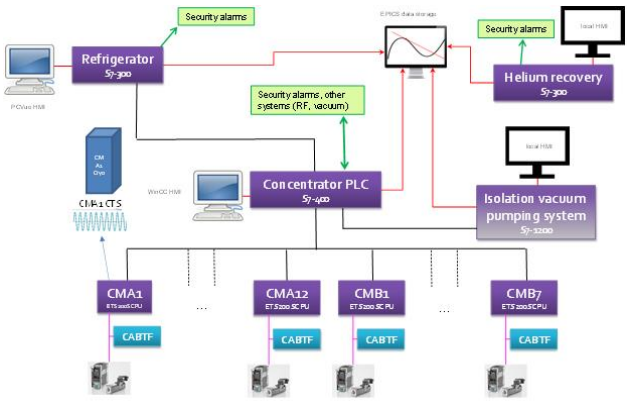


Figure 3: Control system architecture.

are nearly 70 instrumentation boxes for the control of temperatures, pressures and helium levels, vacuum pressures, heating supplies. Orders, sensors feedback and others data are exchanged via the private cryogenic network, which is completely autonomous with independent supervision. During power cuts, batteries make it possible to secure the cryogenic automated system (PLC, supervision and network) and to keep control at all times.

The cryoplant works with its own control system: Siemens PLC and PCVue user interface. Other dedicated Siemens PLCs and their local user interface are used for gas recovery management, for the isolation vacuum pumping system of the Linac, and for the vacuum system of the hot sections and RF cavities

MAIN FUNCTIONS

The functions of the concentrator PLC and19 cryomodules PLC are shown in Fig. 4. The PLCs provide controls independently for each cryomodule by opening or closing the valves and reading process measurements: pressure and level controls, cooldown and warmup regulation. Additionally, the FTS is managed by the same PLC. The orders are made manually by cryogenics operators from the HMI.

Sequential Function Chart

The cryogenics control of the cryomodules is mainly based on sequential steps shown in Fig. 5, each defining a working mode.

- X1: Hot mode, cryomodule is at ambient room temperature.
- X2: checking of the FTS state.
- X20 and X21: waiting cooldown order.
- X22 and X23: the shield and cavity are cooled separately at 60K and 4K. Input valves are opened to immerge the cavity.
- X4: normal mode, the cavity is immerged in the liquid helium bath at 4K. Only this mode allows the activation of the RF fields in the cavity.
- X5: Safety mode: Following a major fault of the cryomodule, helium input is closed and the output valves are completely opened.
- X6: Stand by, the cavity is regulated at a higher temperature to reduce the helium consumption.

Human Machine Interfaces

Three redundant WinCC Pro supervisions are running on virtual machine. Those HMIs are located in a cryogenics

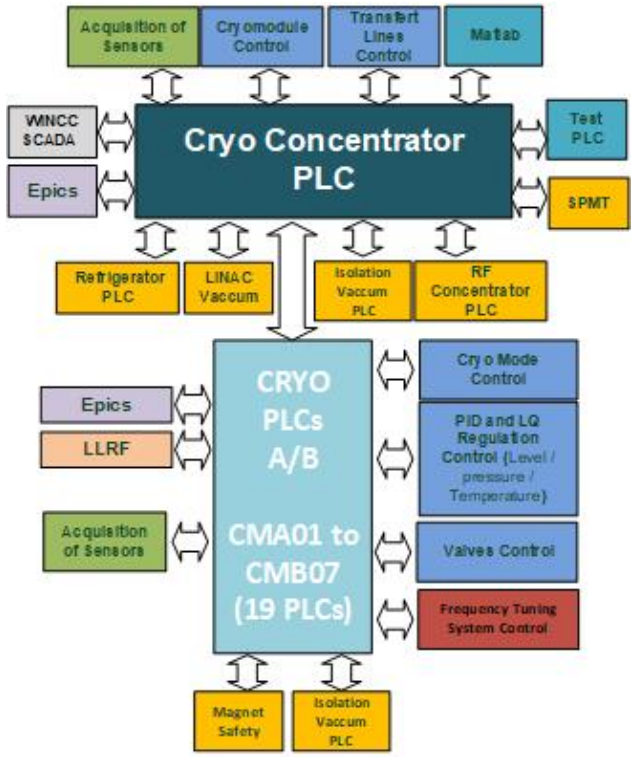


Figure 4: PLCs functions.

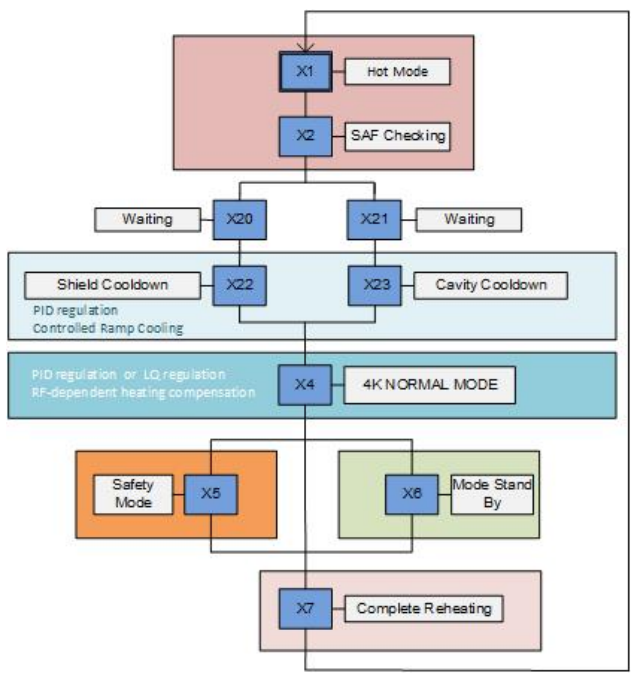


Figure 5: SFC of the cryogenics modes.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

dedicated control room, and are also reachable from distance. They include all the necessary data to drive the cryogenic system, and display the associated commands of the connection box, from transfer lines to cryomodules (see Fig. 6). It is made up of around 200 pages and 6000 variables. The alarms are archived.

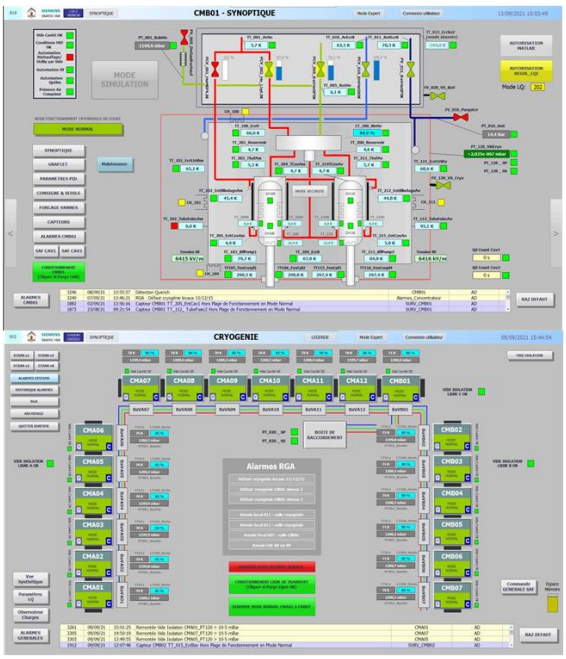


Figure 6: Cryogenics system supervision.

Frequency Tuning System

As the resonance frequency of the superconducting cavity is dependent to its volume, a mechanical device is used to adapt it in real time according to the RF regulation requirements. In the case of type-B cryomodule, the volume is modified through the insertion of a diver inside of the cavity, and in the case of the Type-A, a plier controlled by a screw enable the distortion of the 4 K cold cavity itself. On each cavity, those devices are managed through a single axis motion by the cryomodule PLC, following the phase shift instruction issued by the LLRF. As the deformations of the cavity are only possible at 4 K, hot and cold operating areas are defined (see Fig. 7) according to the cryogenic mode, to prevent any mechanical deterioration as well as any premature wear of the motorization system. In the tuning area, FTS can be positioned around a theoretical tuned position F0, around which it is moved to regulate the RF field.

Simulation

A simulation mode, with a dedicated window on the HMI supervision, was implemented to overwrite commands and sensors feedback on demand. This mode proved itself useful in the commissioning of the cryogenic system, to check every working mode and every safety conditions. This simulator is still used, mainly for the operations of the yearly maintenance phases.

Liquid Helium Bath Control

RF regulation is strongly dependent of the liquid helium bath pressure range (1200 ±2 mbar), and obviously to the immersed state of the cavity. In its earlier version, this requirement is managed by the cryomodule PLC with two PID controllers: the liquid helium bath pressure is regulated through the opening percentage of the output valve (CV005), and its level through the opening percentage of the input valve (CV002). Those controllers working in parallel are managed from a dedicated HMI window (see Fig. 8) where the cryogenics operators can tune the PID parameters and adjust the setpoint.

Moreover, during the cooldown steps, another PID controller regulate the cavity temperature by opening the CV001 valve. The cooling of the Linac is completely controlled with the application of an automatically sliding temperature set point. As shown in Fig. 9, the descent is slowed down from 300K to 150 K and then accelerated from 150K to 4 K in order to limit the time in the critical temperature range 150–50 K.

Dynamic Heat Load Compensation

All control parameters are only valid inside a certain range of conditions, expressed in term of heat load. The cavity RF fields being different for the variety of ions accelerated in Spiral2, the cryogenic bath load will also vary. To avoid permanent optimization of the control parameters, it has been decided to work always at the maximum heat load working point, which is defined as the heat load when the cavity is at its nominal RF field, 6.5 MV/m gradient. Hence, when RF is off, electric heaters are used to mimic the RF load inside

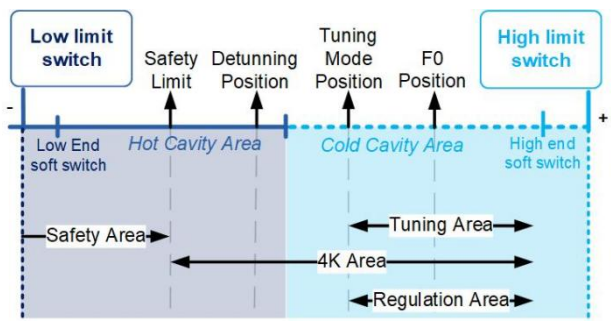


Figure 7: Type-A cavity FTS operating areas.

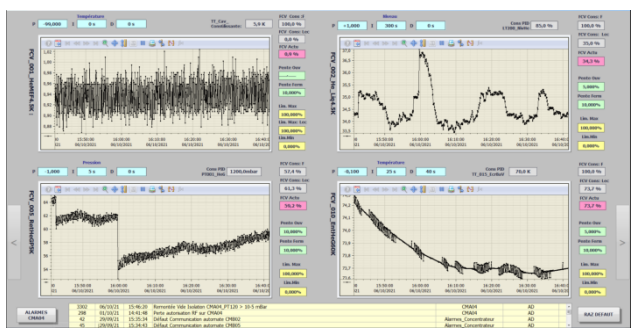


Figure 8: PID parameters supervision.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV006

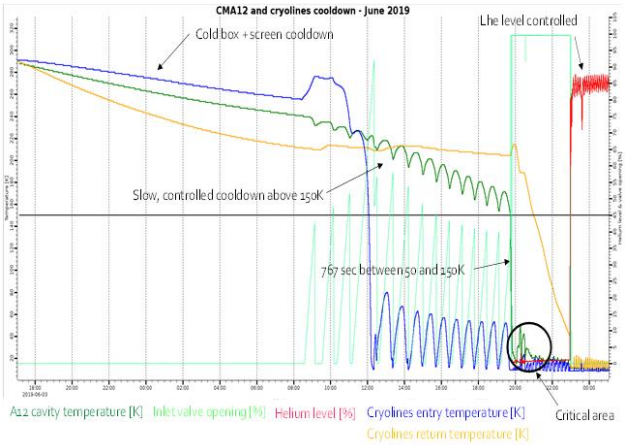


Figure 9: Cooldown of a type-A cryomodule.

the helium baths. Moreover, to keep the liquid helium distribution balanced, the heat load working point is evened up for all the type-A (12.275 W) and type-B (27.47 W) cryomodules. A feedback of the RF field level allow the cryomodule PLC to deliver the correct electrical power to the heater in order to reach those heat load setpoints.

MODEL BASED TOOLS

Cryogenics Modelling

The type-A and type-B cryomodules were modeled using the Simcryogenics [2, 3] library on MATLAB/SIMSCAPE environment. In order to be simple enough to be usable with a PLC, the cryomodule model is linearized to provide the state space equation, linking the opening of the valves and the liquid helium bath state.

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$
$$x = [\rho \; e]^T$$
$$u = [\text{CV002 CV005}]^T$$
$$y = [\text{LT200} \quad \text{PT001}]^T$$

With ρ the density of the liquid helium bath, e the internal energy, CV002 and CV005 the opening of the input and output valves, LT200 and PT001 the helium liquid level and pressure measurements. The state space matrix are computed via a MATLAB graphic interface and transferred from MATLAB to the cryomodules PLCs.

Linear–Quadratic Controller

The linearized model, paired with a disturbance estimator (see Fig. 10), is used to provide a Linear Quadratic (LQ) controller of the liquid helium level and pressure [3].

The feedback gain K and the estimator gain L are computed with MATLAB tools, through the minimization of the quadratic criterion J.

$$J = \int x'Qx + u'Ru$$

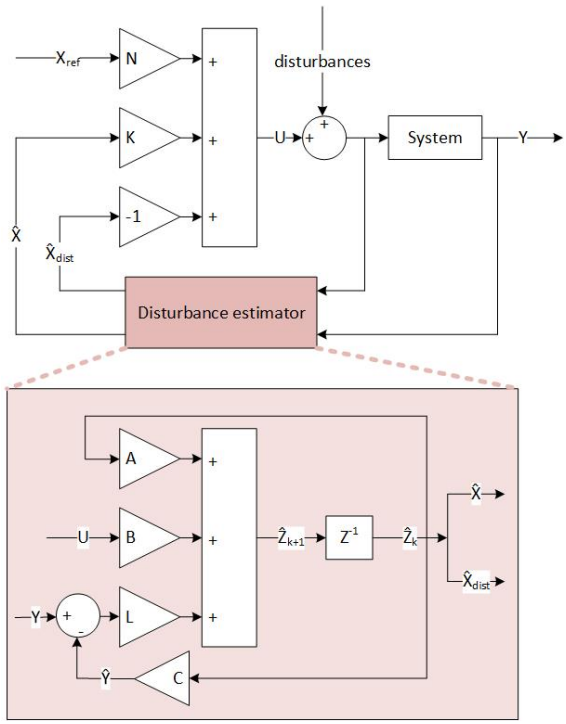


Figure 10: Block diagrams of the LQ controller.

$$Z = [X \quad Y_{dist}]^T$$
$$Z_{k+1} = AZ_k + BU_k + L(Y_k - \hat{Y}_k)$$
$$Y_k = CZ_k$$

The weighting of the variations of the sensors and actuators, as well as the weighting of estimations and errors, are adjustable with a dedicated HMI. The gains are then transferred to the cryomodule PLC with the state space matrix.

This regulation, as it uses knowledge of internal coupling between the liquid helium heat load regarding to the input and output helium flow, shows significantly better performance regarding the pressure stability, especially in the normal 4 K cryogenic mode, than the two separate PID controllers that were firstly used. However, as it is based on a linearization of the modeling around one single working point, its performance is strongly dependent to the distance of the ongoing heat load to this theoretical point. On the contrary the PID regulations can still be robust enough to be used in degrade situation or in the cooldown steps, so it is always possible to manually switch to these controllers.

Heat Load Estimator

The measurement dynamic heat losses, e.g. the variations of the heat load, is an indicator of the cavity state, which could itself be an indicator of the RF losses, or of the beam losses. As it is impossible to have a dynamic measurement of those losses while the accelerator is in nominal operating mode, a state observer was designed [4] (see Fig. 11) and implemented in a dedicated PLC to work in real-time,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

without risks of slowing down the cryomodules PLCs. This estimator PLC communicate with a selected panel of cryomodules PLCs to read the measurement of the CV002 and CV005 valves opening, and liquid helium bath pressure and level.

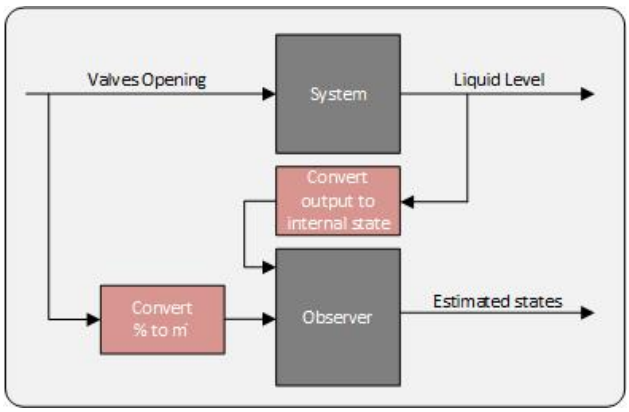


Figure 11: Block diagram of the estimator.

The linearized state space equation is augmented to include the heat load in the state vector. The estimator use a conversion of the input and output valves opening into mass flow, and any estimation error is attributed to a variation of the heat load.

First tests showed that this estimator is able to deliver a slow but correct enough estimation of the heat load, as illustrated by the Estimator 1 in Fig. 12. However it is too dependent on the output valves opening value, which are disturbed by thermo-acoustics oscillations. Others strategies of estimation are being studied to mitigate the weight of those disturbances. For instance the estimator 2 in the Fig. 12 also aims at computing the error of the output mass flow, in addition to the attribution of the error of estimation to the heat load. The result is slower, but, as it is less dependent on the CV005 working point variations, its performance concerning the offset error is improved.

Future improvements on the cryogenic system are expected to mitigate the thermo-acoustic phenomena, and thus to stabilize the CV005 and allow the estimator to provide faster and/or more reliable heat load estimations.

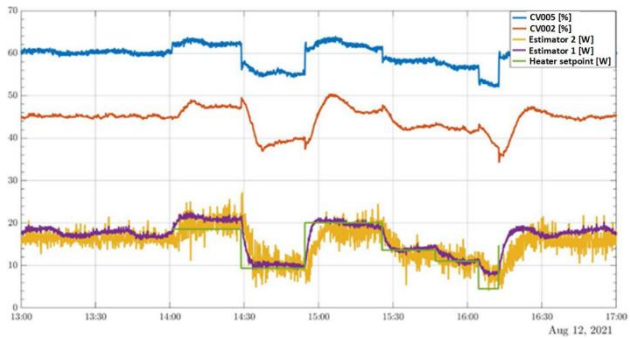


Figure 12: Estimated Heat Load.

In the long term planning, the goal is to design a software sensor able to provide dynamic heat load data from every cryomodule, working with several Kalman and/or Luenberger estimators with different response time. Those data could be useful for the beam tuning and other analysis about the cryogenic system performance.

REFERENCES

[1] P.-E. Bernaudin, "Review of the SPIRAL2 Cryogenic System", GANIL, Caen, France, Rep. ATRIUM-385477, Feb. 2020.

[2] F. Bonne *et al.*, "Modelling and Model-Based-Designed PID Control of theJT-60SA Cryogenic System Using the Simcryogenics Library", *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 171, p. 12028, 2017. doi:10.1088/1757-899X/171/1/012028

[3] A. Vassal, "Study of a multi-client cryogenic system for SPIRAL2", Ph.D. thesis, Phys. University of Caen Normandy, Caen, France, 2019.

[4] A. Vassal *et al.*, "SPIRAL2 Cryogenic System Thermodynamic Behavior Prediction Through Dynamic Modelling", in *Proc. 29th Linear Accelerator Conf. (LINAC'18)*, Beijing, China, Sep. 2018, pp. 366-369. doi:10.18429/JACoW-LINAC2018-TUPO019

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV007

MOTORIZED REGULATION FOR THE SARAF PROJECT

T. Joannem*, D. Darde†, CEA Paris-Saclay, IRFU DIS, Saclay, France
F. Gohier, F. Gougnaud, P. Guiho, A. Roger, N. Solenne, P. Lotrus
CEA Paris-Saclay IRFU DIS, Saclay, France

Abstract

Commissariat à l'Energie Atomique et aux Energies Alternatives (CEA) is in charge of the tuning regulation systems for the SARAF-Linac project. These tuning systems will be used with Low Level Radio-Frequency (LLRF) to regulate three rebuncher cavities and HWR cavities of the four cryomodules. These systems were already tested on the Rebuncher and Equipped Cavity Test Stands (ECTS) to test respectively the warm and cold tunings. This paper describes hardware and software architectures. Both tuning systems are based on Siemens Programmable Logic Controller (PLC) and EPICS-PLC communication. Ambient temperature technology is based on SIEMENS motor controller solution whereas the cold one combines Phytron and PhyMOTION solutions.

INTRODUCTION

Context

SARAF particle accelerator is compose of elements to accelerate, control and tune a beam. To work properly, radio-frequency signals are generate by the Low Level Radio-Frequency (LLRF) system.

These signals are used in many parts of the accelerator to correct the beam. We will focus this document about two of these systems that are using motors to reach this goal: rebunchers and cavities.

First motorize tuning system is locate in the Medium Energy Beam Transfer (MEBT) line. It is part of rebunchers. Second motorize tuning system is locate inside cryomodules, next to each cavity, Fig. 1.

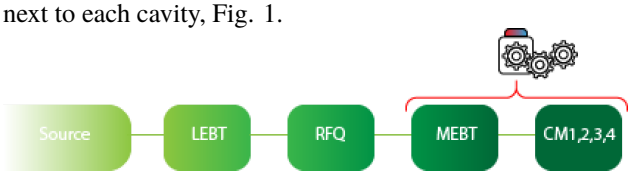


Figure 1: SARAF motors location.

Constraints

Both motorize tuning systems are working in vacuum and radiations environments due to particles accelerator requirements. A third environment parameter have to be taking account and will distinguish two motorizations systems: Temperature.

In fact, motors localize in MEBT, and so rebunchers, will be at ambient temperature therefore cavities motors will be in a cryogenic environment. We will see in next chapters that this difference will heavily impact hardware choice.

* tom.joannem@cea.fr
† david.darde@cea.fr

Goals

Motors have to be define by taking account of working environment but also processes requirements. In our case motors goals are similar but ways to obtain it are different.

In fact, motors have to move or apply a constraints to an element to obtain required frequency of 176 MHz. Motors movements are done according to frequency feedback measured by the LLRF. Using this feedback provide us a way to regulate movements or constraints to obtain required frequency. Rebunchers motors will linearly translate a beam-inside element to get correct frequency and cavities motors will apply mechanical constraints to cavity in order to obtain required frequency.

REBUNCHERS

Presentation

Rebunchers are dynamic tuning devices located on the MEBT section of the SARAF particle accelerator [1], Fig. 2.

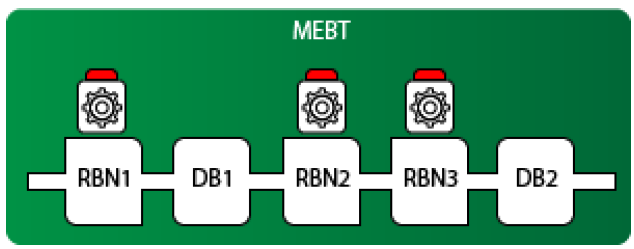


Figure 2: MEBT rebunchers.

They are three similar rebunchers, all located in the MEBT. They are composed of vacuum devices like pumps, gauges and valves but also radio-frequency parts and an outside motorization connected to an internal endless screw that movement radio-frequency passive elements near beam and so gives the possibility to modify feedback frequency.

Movement range of these elements are about thirty millimeters and require a high precision position system and without high-speed response dynamic from the system.

Hardware

According to previously mentioned requirements we will now provide details about hardware used and specifications.

Actuator Rebunchers tuning system requires a position accuracy and speed is not relevant so we choose to use a brushless motor that gives good results for this kind of use.

Moreover due to vacuum high torque requirements motors don't have internal physical brake but are using constant-current to immobilized motor's axis. In fact, motors stator

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

coils are together powered with a constant power and so generate magnetic field that blocks the motor's rotor.

In addition radiations environment near particles beam impose to remove electronics from devices and so to use remote control systems. So motors are without embedded electronic and controllers will be located in remote cabinets outside radiation area, in a maintenance corridor.

Eventually SARAF control system is based on Siemens PLC and so motors are to be compliant to it. To conclude and to respond to the need we choose synchronous motors from Siemens that gives stepper functionality. This Siemens range of hardware is called as Siemens synchronous motors 1FK7, Fig. 3.

Figure 3: Siemens brushless motor.

Position Sensor Because of radiation environment opticals coders can't be use. In fact, incremental and absolute standard position encoders are using optical, and so electronic elements, not compatible with this specific environment.

A position sensor technology compliant with radiation exists: resolvers. These devices don't have embedded electronic and use coils to send impulse electrical signals to a remote acquisition system. The system will receive pulses in the same way than incremental optical coders and will be able to provide a reliable motor angle position.

Resolvers could be embedded with motors, on the rotor axis on remotely connected on a mechanical axis, Fig. 4. In

Figure 4: Siemens resolver.

our case we choose to locate the position sensor on the motor axis to have the best precision possible so it is embedded in the motor and provided as an option.

Conditioning Device Resolver signals are sepcific and requires a conditioning device to convert feedbacks signals to a usable position, Fig. 5.

In fact, resolvers sensors provide three signals: A for first coil pulse, B for second coil pulse and Z or N for a reference signal. Conditioning device have to count pulse according with motor speed acquisition rate to provide position to motor controller.

Figure 5: Resolver conditioning card.

In order to be compliant with resolver and Siemens PLC we used Sinamics sensor module.

Controller Siemens motor controler are composed of two elements: control unit to connect with PLC and power stage according to motor speed rotation and torque connected to motor itself, Fig. 6.

Figure 6: S110 motor controller.

Because we use resolver sensor only one control unit can manage feedbacks from resolver conditioning devices: Siemens S110.

Communication between motor controller and position sensor conditioning device is done with Siemens communication protocol S7. By this way, motor controller can manage motor position and correct it if necessary.

Additionally to this functionality, motor controller is connected to PLC Central Processing Unit (CPU) to receive position commands and to send feedbacks. This communication is done with another Siemens communication protocol based on a Ethernet physical layer: Profinet.

Eventually security motor movement limit switches are connected to controller front end in order to manage motor movement outside normal movement range.

PLC CPU Siemens PLC CPU used on our experiments is the 1516 and provide use the possibility to communicate with motor controller thanks to Profinet and with IOC host via TCP-IP, Fig. 7. CPU is composed of two communication cards and give the possibility to use one card per communication protocol.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV007

Figure 7: Siemens PLC CPU 1516.

In case of power shutdown, position of the motor is memorize in the non-volatile memory of the PLC CPU and so avoid a calibration procedure because of motor position lost.

IOC Host To communicate with EPICS PLC CPU have to be connected to an Input Output Controller (IOC). Communication is managed with TCP-IP and Modbus Protocols. IOC is host on a Industrial PC (IPC).

Global Control Architecture According to previously detailed devices we can now define the global control architecture for rebuncher motorization system.

Software

Frequency Measurement Frequency feedback used for motor position regulation is provided by the LLRF and have to be send to PLC CPU and then to motor controller.

To do it the LLRF writes the frequency value to his own IOC, then it communicates it to PLC CPU IOC that receives it and modifies position command to motor controller according to frequency dynamic.

Regulation Process Frequency regulation process is done using frequency limits. In fact, user defines positive and negative frequency limits, when these limits are reach a motor movement is done to reach frequency target of 176 MHz. Motor movement is divided in five phase: acceleration to high speed, high speed, deceleration from high to low speed, low speed and stop. Two speed profiles have be defined to optimize regulation response time. Low speed is active when measured frequency is lower than 10% of target frequency minus frequency limit. While frequency is within limits motors do not move.

This way to regulate lets the possibility to the operator to manage how precise the regulation should be and also prevents too much motors movement to avoid frequency perturbations created by fast motor movements.

Graphic User Interface EPICS control view provides frequency measurement, motor position, movement limit switches status and more.

From this view automatic frequency regulation parameters could be set, manual motor movement could be done and also motor calibration for maintenance.

CAVITY TUNING UNITS

Presentation

Cavity Tuning Units (CTU) are dynamic tuning devices located on each cavity of each cryomodule of the SARAF particle accelerator. They will be a total of twenty-seven similar systems.

Cryomodule control system manages vacuum, cryogenic, current lead and tuning systems.

Tuning systems are composed of an inside cryogenic environment motorization connected to an internal endless screw that apply a mechanical constraint on related cavity. Modify mechanical characteristic of cavity gives the possibility to modify feedback frequency.

Actuator

Cryogenic environment requires a compatible motor and it remote control. Should be added to cryogenic environment, vacuum and high radiation. To answer to these requirements we use for our experiments hardware based on space applications feedbacks.

We use Phytron stepper motors dedicated for extreme conditions, Fig. 8.

Figure 8: Phytron CSS motors.

We will detailed in next chapters how control architecture of this motor has evolved with our experiments.

Control Architecture Based on Siemens 300 PLC Generation

First control architecture was done with previous generation of Siemens PLC cards. Indeed, Phytron provides a control card named "1-STEP-Drive" that fits with Siemens ET200S remote input-output cards, Fig. 9.

This control architecture was with last generation Siemens 1516 CPU and previous generation ET200S cards. In this configuration motor did not include embedded resolver.

Control Architecture Based on Siemens 1500 PLC Generation

Second control architecture was done with last genera-tion of Siemens PLC cards. Indeed, Phytron provides a control

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 9: ET200S and Phytron controller.

card named "TM StepDrive 1x24..48V/5A" that fits with Siemens ET200SP remote input-output cards, Fig. 10.

Figure 10: ET200SP control architecture.

This control architecture was with last generation Siemens 1516 CPU and last generation ET200SP cards. In this configuration motor did not include embedded resolver.

Control Architecture Based on Siemens 1500 PLC Generation with Position Feedback

This architecture is similar to previous control architecture with additional hardware. In fact motor includes embedded resolver and a fast acquisition digital input card was added on the CPU rack, Fig. 11. We could get resolver position and confirm that motor move according to positions commands.

Figure 11: Fast acquisition card.

In previous configurations we controlled motor with an open-loop control architecture but with the resolver implantation we could detect any issue thanks to a close loop. Previously the only way to detect that motor did not move even when command was send was to verify feedback frequency. It should change according to mechanical constraints on cavity, if not motor is considered as blocked.

This architecture was tested on the Equipped Cavity Test Stand (ECTS) after a motor block event due to an issue on the cavity constraints endless screw.

Control Architecture Based on PhyMOTION Control

Control architecture of Phytron motors for SARAF cryomodules is no longer based on dedicated Siemens compatible control cards but with a Phytron modular control system, Fig. 12.

Figure 12: PhyMotion controller.

The PhyMOTION is composed of power card, communication card, like for Profinet, and control cards. We managed SARAF cryomodules PLC-based control architecture to be similar for each cryomodule. In fact one PLC CPU manage every cryomodule features for one cryomodule and we apply the same philosophy for the PhyMOTION.

Cryomodules are composed from six to seven cavities each one connected to a Phytron motor. Each cryomodule have is own PhyMOTION controller composed of motor control card, one per motor. These cards can manage motors power but also position feedback from resolver.

SARAF project strategy was not to use resolver to get a close loop regulation system but to use frequency measurement

FUTURE APPLICATIONS

Control Architecture Based on PhyMOTION Control with Position Feedback

Thanks to SARAF project feedback about PhyMOTION integration we will be able to deploy a close loop control with cryogenic motors in our next experiments.

Motor with embedded resolver will give us the possibility to improve reliability without depending of an external measurement source.

CONCLUSION

Technology for motor command and position feedback into radiation, vacuum and even cryogenic environment improves and provide use user friendly hardware and software solutions that can be connected with industrial standards.

Embedded resolver gives the possibility to have a local close loop control and improve diagnostics for non-accessible devices.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV007

ACKNOWLEDGEMENTS

We would like to thanks every people that help us and contribute to motors integration study, deployment and test in both CEA department DIS and DACM.

REFERENCES

[1] Lu Zhao, "Study and test of the rebuncher for SARAF-LINAC phase 2", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 986, p. 164716, 2021. doi:10.1016/j.nima.2020.164716

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

OpenCMW - A MODULAR OPEN COMMON MIDDLE-WARE LIBRARY FOR EQUIPMENT- AND BEAM-BASED CONTROL SYSTEMS AT FAIR

Ralph J. Steinhagen, H. Bräuning, D. Day, A. Krimm, T. Milosic,
D. Ondreka, A. Schwinn, GSI Helmholtzzentrum, Darmstadt, Germany

Abstract

OpenCMW is an open-source modular event-driven micro- and middle-ware library for equipment- and beam-based monitoring as well as feedback control systems for the FAIR Accelerator Facility.

Based on modern C++20 and Java concepts, it provides common communication protocols, interfaces to data visualisation and processing tools that aid engineers and physicists at FAIR in writing functional high-level monitoring and (semi-)automated feedback applications.

The focus is put on minimising the required boiler-plate code, programming expertise, common error sources, and significantly lowering the entry-threshold that is required with the framework. OpenCMW takes care of most of the communication, data-serialisation, data-aggregation, settings management, Role-Based-Access-Control (RBAC), and other tedious but necessary control system integrations while still being open to expert-level modifications, extensions or improvements.

ARCHITECTURE

OpenCMW is a light-weight modular middle-ware twin-library that combines ØMQ, REST and micro-service design patterns illustrated in Fig. 1. The Majordomo Protocol Broker (MDP) provides reliable (a)synchronous request-reply as well as publish-subscribe (and related radio-dish) communication patterns between external clients and workers, implementing the business logic that may reside either internally or externally to the MDP process. Its core relies primarily on the high-performance and low-latency ØMQ-based transport layer but can, for example, be optionally extended by HTTP to also support REST or HTML-based communication patterns, and optionally secure worker access via RBAC [1–16].

OpenCMW strongly embraces lean-code principles and hence aims at minimising boiler-plate code and to aid light-weight development of network-based, semi- to fully automated real-time feedback applications. It thus provides template implementations for common tasks such as to

a) aggregate, synchronise, and to sanitise data received from multiple devices,

b) allow to inject custom domain-specific user-code to numerically post-process the received data, and

c) derive control signals and forward these to other services using common communication libraries.

This business logic is implemented by workers that cover:

- optional class-based domain-object definitions for the input parameter and return value, and
- two event-handler interfaces that are registered either with the MDP or Event Store, and that implement the call-back functions further described below.

Notably, developers do not need to rely on IDL-type or other proprietary definitions that, based on our experience, may become hard to synchronise and maintain in later development iterations. The interfaces are defined by standard POCO or POJO domain-objects that OpenCMW analyses using `constexpr` compile-time (C++) or run-time (Java) reflection, further described below.

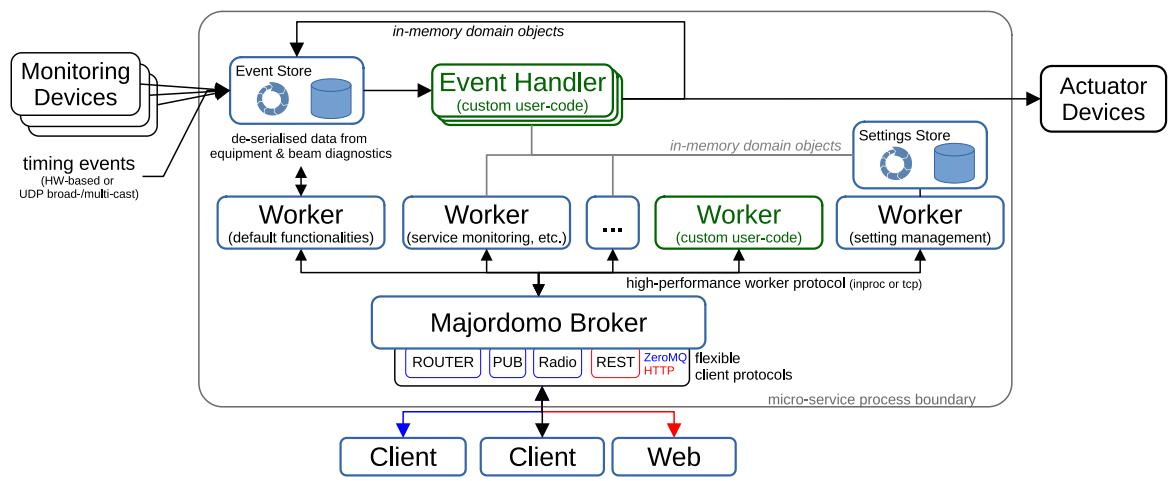


Figure 1: OpenCMW micro-service system architecture combining the Majordomo and event-sourcing design pattern.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

DEVICE MODELLING

FAIR provides a wide variety of particle beams with different properties and extraction parameters that are transported to multiple experiments or storage rings either time-multiplexed or even simultaneously. Accelerator operation is thus organised into semi-repetitive 'patterns' and nested substructures (aka. 'timing contexts' or '<ctx>') that are triggered by timing events in order to orchestrate the required rapid real-time succession of individual actions, data acquisition, or device reference settings changes.

OpenCMW is flexible with its domain-object modelling and supports arbitrary setting-types from single scalar fields up to complex nested tree-like data-structures. However, based on operational experience at GSI, CERN and other accelerator facilities, and to be compatible with the existing implementations, most equipment is modelled using a reduced flat 'device/property' scheme only [21]. This scheme models settings essentially through ASCII-based endpoints, that include filters and loosely follows the more commonly-known RFC 1808 URI definition [22]:

```
<device>/<property>?ctx=<ctx>;
  <filter_i=val_j>; ... ;<filter_n=val_m>
```

with the ordering of the optional context selector and filters after the initial '<device>/<property>' being arbitrary and separated by a semi-colon (';') or ampersand ('&').

DATA AGGREGATION

Most monitoring and feedbacks are rarely SISO but more commonly MIMO systems. OpenCMW thus provides facilities that aid in the aggregation and synchronisation of input data that may arrive through different transport channels, protocols, and from a large number of equipment- or beam-based systems. To process this data efficiently, as well as to simplify and reduce potential error sources for developers, two aggregation strategies are provided:

a) continuous sample-by-sample or chunked data not aligned to a <ctx>: these are typically processed by a GNU-Radio-based flow-graph, commonly used in the domain of software-defined-radios (SDRs [17]), as illustrated in Fig. 2;

b) chunked data that is already (partially) aligned to a timing <ctx>: these are typically processed using the event-sourcing pattern that combines the data across multiple input sources or larger superordinate <ctx> or to perform signal post-processing routines on a chunk-by-chunk basis as illustrated in Fig. 3.

The event-sourcing relies on different types of workers:

- 'adapter': tasked with receiving and de-serialising the specific (possibly foreign) wire-format, and storing the resulting domain-object alongside its meta-information into the event store's event stream;
- '<ctx>-matcher': collecting multiple source domain-objects into one combined new 'aggregate' which is usually triggered by timing events or arrival of a new <ctx>. The new aggregate is stored inside the same (or optionally another) event store event/stream, either once all required data arrived or once a configurable time-out w.r.t. the aggregation start is reached;
- post-processing workers: performing the actual post-processing and control actions based on user-supplied worker handler-callback code;

The handler-callbacks are triggered whenever a new aggregate is written to the event store, receive the actual and past issuing events, data to facilitate FIR- and IIR-type filtering, and may notify the MDP to publish the intermediate results.

DESIGN CONSIDERATIONS

There have been many expensive software failures, accidents, and inefficiencies in the past, most famously the Ariane flight 501 and Mars Climate Orbiter that failed due to avoidable type- and physical-unit conversion errors [23, 24]. Our accelerator domain is not immune to these type of failures, and struggles in addition with organically grown code-bases that often – as a consequence of Conway's law[1], time- and resource constraints – cannot be adequately pruned, refactored or optimised. OpenCMW thus aims at lean, modern, and long-term sustainable architecture that avoids the above mistakes by design.

[1] "[Developers will] produce a design whose structure is a copy of the organization's communication structure", [25]

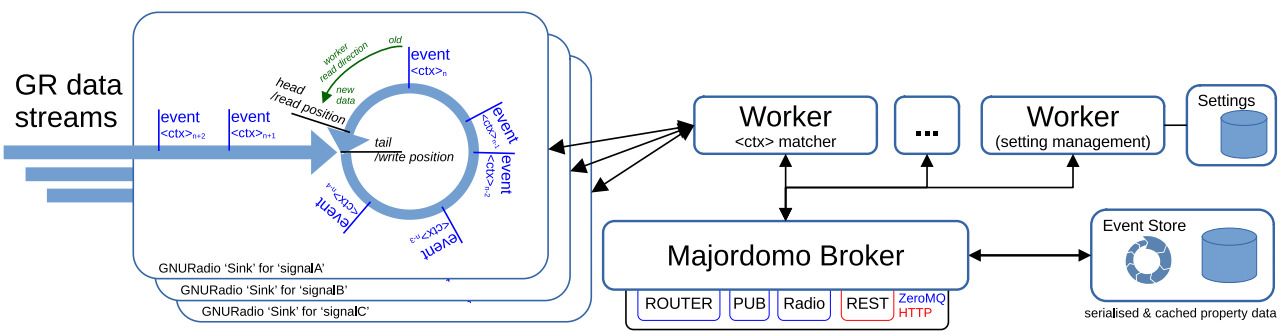


Figure 2: Schematic GNU-Radio-based (GR) processing of continuous data as used by the FAIR Digitizers [17–20]. The data processed by the flow-graph is stored in non-blocking circular buffers (sinks) similar to tagged b-trees that allow to efficiently chunk the data according to the required timing <ctx> scope.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

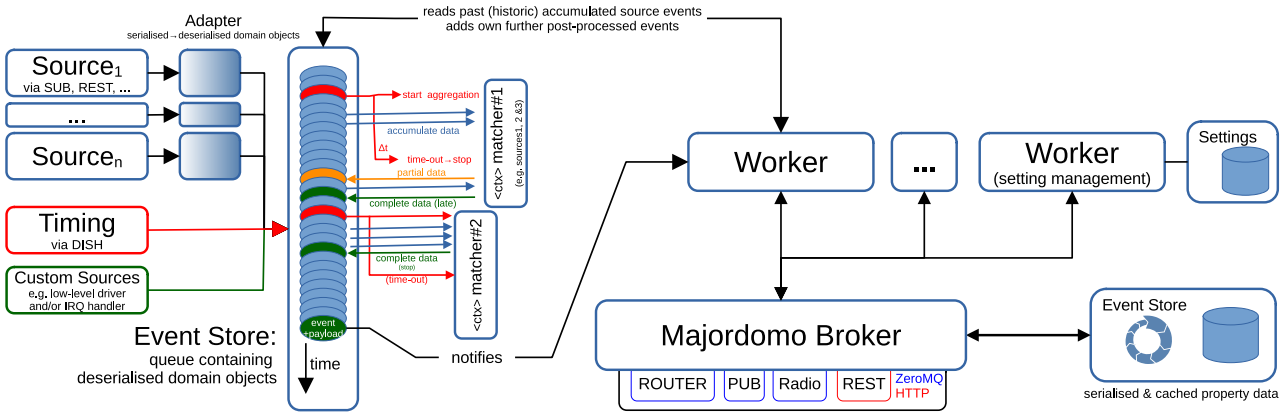


Figure 3: Schematic <ctx> multiplexed-based data aggregation based on the event-sourcing design pattern [11].

The transport layer and data serialisation are basic core functionalities of OpenCMW or any middle-ware for transmitting, storing and later retrieving information by (often quite) different subsystems. With a multitude to choose from, a non-negligible subset of frameworks claim to be the fastest, most efficient, easiest-to-use, etc. A fair comparison of their performance is rather complex, highly non-trivial, and thus subjective because the underlying assumptions of 'what counts as important' is quite different between specific domains.

Instead, we want to document the rationale as to why we decided to implement a new middle-ware, its considerations, constraints, technical core choices, and goals that went into OpenCMW's design in the hope that it might be interesting, perhaps be adopted, inspire new ideas, or any other form of improvement. OpenCMW's core paradigms, loosely ordered according to their importance, are:

1. undogmatic lean and agile software design principles.
2. performance: end-to-end real-time latency minimisation between the service's data object being ready, serialised, and sent, until it is received, fully de-serialised, and ready for further processing by the client[2].
3. loose coupling between server- and corresponding client-side-definitions to provide soft sustainable upgrade paths.
4. strict separation of concern between the communication transport layer, serialiser protocol, and user supplied custom-code: OpenCMW provides extendable interfaces to transparently exchange or add new transport and serialisation protocols simplifying the custom user-code that does not need to be aware of the actual protocols that the client may request. This enables more modular implementation, easier maintenance, and efficient code-sharing of code blocks between different projects.
5. static compile-time reflection[3][26, 27]: derive interfaces directly from C++ or Java domain-objects:
   - less error-prone and more efficient compared to other IDL-based solutions that need to rely on external code-generation tools.
   - greatly reduces the cognitive complexity, see List. 1 and the appendix for code examples.
   - improves productivity of new, occasional, or less-experienced developers who need to be only "vaguely" familiar with C++ or Java.
   - supports UTF8, all primitive/fundamental data types, nested objects, common data containers.
   - efficient (first-class) support of large collections of numeric (floating-point) data.
6. strong type- and physical-units safety [28–30].
7. OpenAPI [31]: self-documented data-structures with optional `constexpr` data field annotations to communicate the service's data-exchange-API intent to the client (e.g. physical unit, access constraints, human-understandable description, etc.). This meta-information can be used to re-generate the domain-objects on the client-side.
8. Modern C++20 [32] and Java 11 standards:
   - more concise with respect to expressing developer's intent and constraints.
   - easier, safer, and more intuitive syntax for new or occasional developers.
   - better performance, due to larger proliferation of compile-time `constexpr` and `consteval` expressions and library functionalities.
9. minimise external library dependencies: especially rely on those already or targeted to become official part of the C++ ISO standard [26, 28, 32].
10. minimise code-base and code-bloat while providing hard-to-implement core-functionalities: small code fits more likely into CPUs's cache and thus results in faster execution. Also from a maintenance perspective, more code requires more time to read and understand, and is harder to modify, test, or fix.

[2] Network-IO bandwidth limit driven trade-offs between size and en-/decoding speed are not of primary concern for our use-case
[3] N.B. Java is presently limited to run-time reflection.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

```cpp
struct CppDomainObjectExample {
  Annotated<float, thermodynamic_temperature<kelvin>, "device specific temperature">        temperature     = 23.2F;
  Annotated<float, electric_current<ampere>, "this is the current from ...">                current         = 42.F;
  Annotated<float, energy<electronvolt>, "SIS18 energy at injection before being captured"> injectionEnergy = 8.44e6F;
  std:string                                                                                notAnnotated    = "Hello World!";
  // [..]
};
// refl-cpp-based: targeted and becomes obsolete with the next C++ standard
ENABLE_REFLECTION_FOR(CppDomainObjectExample, temperature, current, injectionEnergy, notAnnotated)
```

```java
public class JavaDomainObjectExample {
  @MetaInfo(description = "device specific temperature", unit = "K")
  public float   temperature     = 23.2f;
  @MetaInfo(description = "this is the current from ...", unit = "A")
  public float   current         = 42.f;
  @MetaInfo(description = "SIS18 energy at injection before being captured", unit = "eV")
  public float   injectionEnergy = 8.44e6f;
  public String  notAnnotated    = "Hello World!";
  // [..]
}
```

Listing 1: C++ POCO and equivalent Java POJO domain-object reflection example.

11. unit-test and CI/CD driven development to minimise errors and detect potential (also quantitative performance) regressions.
12. free- and open-source LGPLv3-licensed code-base.

It is important to us that this code can be re-used, built- and improved-upon by anybody and thus we hope that the above paradigms lower the entry-level and improve the likelihood of OpenCMW to be adopted also by external users that wish to understand, upgrade, or bug-fix 'what is under the hood' or what is of specific interest to them.

PERFORMANCE

In real-time control and notably for feedback systems, performance depends not only on correct results but also when it is applied to the accelerator or presented to the operators. The required maximum latency that is to be minimised, corresponding group-delay and phase-lag depends on the specific application. OpenCMW's primary use-case are high-level software-based monitoring and (semi-)automated feedback applications at FAIR with bandwidths of up to 25 Hz and latencies well below 10 ms. The framework itself should contribute only with a small proportion to these latencies.

Since the OpenCMW architecture by design strongly decouples, and since there are a wide range of possible combinations, we independently benchmarked the transport-layer only using small dummy-payloads and serialisation without the transport layer.

Majordomo Broker

The performance results for our Java-based Majordomo implementation for both MDP in-process (InProc) and external worker (TCP, via *localhost*) is shown in Table 1. The performance is largely determined by the specific ØMQ implementation (*JeroMQ*) in the above case [33], polling loop optimisation, and by itself contributes very little to the overall latency. This is deemed sufficient for our few Java-based applications with lower latency requirements. The C++-based Majordomo implementation, based on *libzmq* and loosely on the MDP reference implementation [34, 35],

Table 1: OpenCMW-Java Majordomo Performance (Test-System: AMD Ryzen 9 5900X)

ØMQ Transport Type	msgs/s
REQ/REP synchronous via TCP	12594
REQ/REP synchronous via InProc	27247
REQ/REP asynchronous via TCP	285714
REQ/REP asynchronous via InProc	400000
SUBSCRIBE via SUB-Socket	1538461
SUBSCRIBE via DEALER-socket	666666

aims at latencies below 1 ms to be compatible with FAIR's SDR applications [17–20]. This is presently being optimised and will be released soon.

Serialiser

OpenCMW aims to provide interfaces for service-client communication where: a) high-performance is important – usually relying on binary-type (de-)serialisation wire-formats – and b) where ease of access, notably, REST and Web-based compatibility is important, often relying on string-based wire-formats such as JSON, YAML, XML, etc. Figure 4 shows the round-trip performance results for OpenCMW and various other common implementations for both C++ and Java [1, 2, 36–42].

While some implementations provided only map-style interfaces, each serialiser has been extended and optimised to (de-)serialise the same equivalent domain-object back-to-back to have more comparable results with the compile-time reflection-based approach. A large spread within but also between binary- and string-based wire-format encoding libraries is visible. The 'CmwLight' serialiser is OpenCMW's functional open-source re-implementation of the existing proprietary CMW-Data protocol that is presently used internally at GSI[21]. The benchmark is available at [43]. Pull-Requests to improve the OpenCMW and other serialisers are welcome. Tentative results indicate that many of the serialiser supporting binary wire-formats seem to be optimised for simple data structures that are typically much smaller than 1k Bytes rather than large numeric ar-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

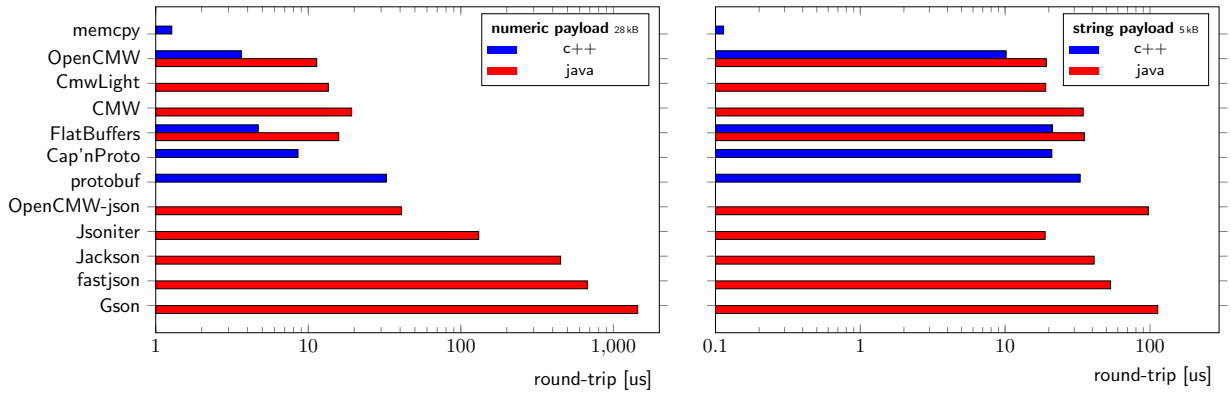


Figure 4: Round-trip performance for converting the input- to output-domain-object via the given wire-format for numeric- (left) and `std::string`-data (right) dominated domain-objects (N.B. log-scale, smaller is better).

rays that were either slow to encode and/or required custom serialiser extensions. *Jsoniter* showed the best results for parsing JSON in Java [39]. We thus decided to use it as the parsing back-end for the *OpenCMW-JSON* deserialiser also to avoid common mistakes and error-handling when processing incomplete or erroneous JSON streams that the deserialiser must be able to handle. Since these error types are non-issues for the serialiser side, we adapted a custom JSON serialiser implementation to the compile-time reflection approach which proved to be more performant. This is also the reason for the visible performance differences between *OpenCMW-JSON* and *jsoniter* in Fig. 4. At the time of the test, the C++ *simdjson* library was not yet evaluated but appears promising and will thus be included in future benchmarks [44].

DOMAIN API EVOLUTION

An important consideration for every distributed system is how to be able to update the API defining domain-objects without having to simultaneously update every other component of the system. To allow for a reliable update strategy OpenCMW uses the transferred metadata to validate the data during deserialisation.

The MDP broker (or any other caller of the serialiser) can control if metadata should be serialised, the type of checks that should be performed on deserialisation, and how errors should be treated. The behaviour of the error handling can be controlled with domain-object annotations.

To optimise the performance impact for high update rates, OpenCMW assumes domain objects stay identical during one connection and only performs checks on the first update. The MDP client calls the verbose deserialiser on the first update and the fast deserialiser on subsequent updates. If there are differences in the first update it can:

- ignore compatible changes like new optional fields or missing optional fields. The client code can still check for the presence of optional fields through the return value of the serialiser.
- request a fallback property if available and warn the user of the legacy API.
- throw an exception.

Worker implementations should keep future extensions in mind when defining their domain objects, but not at the cost of a clean design. When changes are necessary, new (non-mandatory) fields can be added in a backward compatible way if they don't change the meaning of the other data. If non-compatible changes like changing data types are performed, a fallback property providing the old domain object should be offered in the 'two on production' pattern[45, 46]. It is encouraged to also make use of the 'aggressive obsolescence' and 'experimental preview' patterns, to keep the burden of API maintenance manageable. There is no explicit versioning of the domain objects, mismatches in domain objects are determined by the deserialiser.

CONCLUSION

OpenCMW is a new open-source modular event-driven micro-service and middle-ware library for equipment- and beam-based monitoring as well as feedback control systems for the FAIR Accelerator Facility.

Based on modern C++20 and Java concepts, together with the benefits and chances of an agile open-source development, we believe that the above lean software development principles are key ingredients and will help experienced as well as entry-level engineers and physicists to write, share, and improve functional and long-term maintainable code for high-level monitoring and (semi-)automated feedback applications both at FAIR, other laboratories as well as used by other interested developers elsewhere. For detailed information, code examples, or to participate in this project, please visit and interact with the authors via [1–3].

ACKNOWLEDGEMENTS

The implementation heavily relies upon and re-uses time-tried and well-established concepts from ØMQ (notably the Majordomo communication pattern [8]), LMAX's lock-free ring-buffer disruptor [11], GNU-Radio real-time signal processing framework [17], Javalin for the RESTful interface [47], as well as previous implementations and experiences gained at GSI, FAIR and CERN [21, 48–57].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

APPENDIX

```java
package io.opencmw.server.rest.samples;

import static io.opencmw.OpenCmwProtocol.EMPTY_URI;

import java.io.IOException;
import java.net.URI;
import java.util.Objects;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.TimeUnit;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.zeromq.ZContext;

import io.opencmw.domain.NoData;
import io.opencmw.rbac.BasicRbacRole;
import io.opencmw.rbac.RbacRole;
import io.opencmw.serialiser.annotations.MetaInfo;
import io.opencmw.server.MajordomoBroker;
import io.opencmw.server.MajordomoWorker;
import io.opencmw.server.rest.MajordomoRestPlugin;

import zmq.util.Utils;

public class BasicSample {
    private static final Logger LOGGER = LoggerFactory.getLogger(BasicSample.class);

    public static void main(String[] argv) throws IOException {
        final MajordomoBroker broker = new MajordomoBroker("PrimaryBroker", EMPTY_URI, BasicRbacRole.values());
        final URI brokerRouterAddress = broker.bind(URI.create("mdp://*:" + Utils.findOpenPort()));
        final URI brokerSubscriptionAddress = broker.bind(URI.create("mds://*:" + Utils.findOpenPort()));
        broker.start();
        new MajordomoRestPlugin(broker.getContext(), "Test HTTP/REST Server", "*:8080").start();
        // instantiating and starting custom user-service
        new HelloWorldWorker(broker.getContext(), "helloWorld", BasicRbacRole.ANYONE).start();
    }

    @MetaInfo(description = "My first 'Hello World!' Service")
    public static class HelloWorldWorker extends MajordomoWorker<BasicRequestCtx, NoData, ReplyData> {
        public HelloWorldWorker(final ZContext ctx, final String serviceName, final RbacRole<?>... rbacRoles) {
            super(ctx, serviceName, BasicRequestCtx.class, NoData.class, ReplyData.class, rbacRoles);

            // the custom used code:
            this.setHandler((rawCtx, requestContext, requestData, replyContext, replyData) -> {
                final String name = Objects.requireNonNullElse(requestContext.name, "");
                LOGGER.atInfo().addArgument(rawCtx.req.command).addArgument(rawCtx.req.topic)
                    .log("{} request for worker - requested topic '{}'");
                replyData.returnValue = name.isBlank() ? "Hello World" : "Hello, " + name + "!";
                replyContext.name = name.isBlank() ? "At" : (name + ", at") + " your service!";
            });

            // simple asynchronous notify example - (real-world use-cases would use another updater than Timer)
            new Timer(true).scheduleAtFixedRate(new TimerTask() {
                private final BasicRequestCtx notifyContext = new BasicRequestCtx(); // re-use to avoid gc
                private final ReplyData notifyData = new ReplyData(); // re-use to avoid gc
                private int i;
                @Override
                public void run() {
                    notifyContext.name = "update context #" + i;
                    notifyData.returnValue = "arbitrary data - update iteration #" + i++;
                    try {
                        HelloWorldWorker.this.notify(notifyContext, notifyData);
                    } catch (Exception e) {
                        // further handle exception if necessary
                    }
                }
            }, TimeUnit.SECONDS.toMillis(1), TimeUnit.SECONDS.toMillis(2));
        }
    }

    @MetaInfo(description = "arbitrary request domain context object", direction = "IN")
    public static class BasicRequestCtx {
        @MetaInfo(description = " optional 'name' OpenAPI documentation")
        public String name;
    }

    @MetaInfo(description = "arbitrary reply domain object", direction = "OUT")
    public static class ReplyData {
        @MetaInfo(description = " optional 'returnValue' OpenAPI documentation", unit = "a string")
        public String returnValue;
    }
}
```

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] FAIR. 'OpenCMW C++ Project'. (2021), https://github.com/fair-acc/opencmw-cpp

[2] FAIR. 'OpenCMW Java Project'. (2020), https://github.com/fair-acc/opencmw-java

[3] FAIR. 'OpenCMW Gitter Forum'. (2021), https://gitter.im/fair-acc/opencmw

[4] A. Krimm and R. Steinhagen, 'FAIR Common Specification - Modular Open Common Middle-Ware Library for Equipment- and Beam-Based Control Systems of the FAIR Accelerators', FAIR, Tech. Rep., 2020. https://edms.cern.ch/document/2444348

[5] P. Hintjens, *ZeroMQ*. O'Reilly Media, Inc., 2013, ISBN: 9781449334062. https://zguide.zeromq.org

[6] ZeroMQ, 'ZeroMQ home-page', The ZeroMQ Project, Tech. Rep., 2020. https://zeromq.org/

[7] ZeroMQ, 'ZeroMQ Project Respository', The ZeroMQ Project, Tech. Rep., 2013-2020. https://github.com/zeromq

[8] P. Hintjens, 'Majordomo Protocol RFC', The ZeroMQ Project, Tech. Rep., 2012. https://rfc.zeromq.org/spec/18/

[9] P. Hintjens, 'ZeroMQ Publish-Subscribe RFC', The ZeroMQ Project, Tech. Rep., 2014. https://rfc.zeromq.org/spec/29/

[10] D. Somech, 'ZeroMQ Radio-Dish RFC', The ZeroMQ Project, Tech. Rep., 2020. https://rfc.zeromq.org/spec/48/

[11] M. Thompson, D. Farley, M. Barker, P. Gee and A. Stewart. 'LMAX Disruptor: High performance alternative to bounded queues for exchanging data between concurrent threads'. (2011), https://lmax-exchange.github.io/disruptor/

[12] National Institute of Standards and Technology, 'Role Based Access Control - RBAC', NIST, Tech. Rep., 2020. https://csrc.nist.gov/projects/role-based-access-control/faqs

[13] D. F. Ferraiolo, D. R. Kuhn and R. Chandramouli, *Role-Based Access Control*, 2nd. USA: Artech House, Inc., 2007, ISBN: 1596931132.

[14] R. T. Fielding, 'Architectural Styles and the Design of Network-based Software Architectures - Chapter 5: Representational State Transfer (REST)', Ph.D. dissertation, University of California, Irvine, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[15] T. Berners-Lee and D. Connolly, *Hypertext markup language — 2.0*, Internet RFC 1866, Nov. 1995.

[16] R. Fielding *et al.*, 'RFC 2616, Hypertext Transfer Protocol – HTTP/1.1', RFC Editor, Tech. Rep., 1999. http://www.rfc.net/rfc2616.html

[17] GNU Radio Website. (accessed May 2020), http://www.gnuradio.org

[18] R. J. Steinhagen *et al.*, 'Generic Digitization of Analog Signals at FAIR – First Prototype Results at GSI', in *Proc. 10th International Particle Accelerator Conference (IPAC'19), Melbourne, Australia, 19-24 May 2019*, (Melbourne, Australia), ser. International Particle Accelerator Conference, Geneva, Switzerland: JACoW Publishing, Jun. 2019, pp. 2514–2517, ISBN: 978-3-95450-208-0. DOI: 10.18429/JACoW-IPAC2019-WEPGW021. http://jacow.org/ipac2019/papers/wepgw021.pdf

[19] FAIR. 'gr-digitizers repository'. (2021), https://github.com/fair-acc/gr-digitizers

[20] FAIR. 'gr-flowgraph repository'. (2021), https://github.com/fair-acc/gr-flowgraph

[21] J. Lauener and W. Sliwinski, 'How to design & implement a modern communication middleware based on ZeroMQ', in *Proceedings of ICALEPCS'2017*, (Barcelona, Spain), JACoW, Ed., ser. International Conference on Accelerator and Large Experimental Control Systems, 2017, MOBPL05. 7 p. DOI: 10.18429/JACoW-ICALEPCS2017-MOBPL05. https://cds.cern.ch/record/2305650

[22] R. T. Fielding, 'Relative Uniform Resource Locators', RFC Editor, Tech. Rep. 1808, Jun. 1995, 16 pp. DOI: 10.17487/RFC1808. https://rfc-editor.org/rfc/rfc1808.txt

[23] J.-L. L. et al, *ARIANE 5 Flight 501 Failure: Report by the Enquiry Board*. European Space Agency — ESA, Jul. 1996. https://esamultimedia.esa.int/docs/esa-x-1819eng.pdf

[24] A. G. Stephenson *et al.*, *Mars Climate Orbiter Mishap Investigation Board: Phase I Report*. Jet Propulsion Laboratory (U.S.), United States. National Aeronautics and Space Administration, Nov. 1999. https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf

[25] M. E. Conway, 'How do committees invent?', *Datamation*, Apr. 1968. https://www.melconway.com/Home/pdf/committees.pdf

[26] D. Sankel. 'N4856 – C++ Extensions for Reflection'. (Mar. 2020), http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/n4856.pdf

[27] V. Karaganev. 'refl-cpp — a compile-time reflection library for C++'. (2020), https://github.com/veselink1/refl-cpp

[28] M. Pusz. 'P1935R2 – A C++ Approach to Physical Units'. (Jan. 2020), http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/p1935r2.html

[29] M. Pusz. 'A Physical Units Library For the Next C++'. (Sep. 2020), https://github.com/CppCon/CppCon2020/blob/main/Presentations/a_physical_units_library_for_the_next_cpp

[30] M. Pusz. 'mp-units - A Units Library for C++'. (2021), https://github.com/mpusz/units

[31] OpenAPI Initiative. 'The OpenAPI Specification'. (2016-2021), https://github.com/OAI/OpenAPI-Specification

[32] ISO, *ISO/IEC 14882:2020: Programming languages — C++*. Geneva, Switzerland: International Organization for Standardization — ISO, 2020, p. 1853. https://www.iso.org/standard/79358.html

[33] ZeroMQ. 'JeroMQ – Pure Java implementation of libzmq'. (2021), https://github.com/zeromq/jeromq

[34] ZeroMQ. 'libzmq – ZeroMQ core engine in C++, implements ZMTP/3.1'. (2021), https://github.com/zeromq/libzmq

[35] ZeroMQ. 'Majordomo Reference Implementation'. (2021), https://github.com/zeromq/majordomo

[36] Renshaw, David and Hancock, Harris and Varda, Kenton. 'Cap'n Proto Library'. (2021), https://github.com/capnproto/capnproto

[37] W. van Oortmerssen et al. 'FlatBuffers'. (2021), https://github.com/google/flatbuffers

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV009

[38] S. Ghemawat, J. Dean, D. Dulitz, C. Silverstein, P. Haahr and C. Anderson. 'Protocol Buffers - Google's data interchange format'. (2021), https://github.com/protocolbuffers/protobuf

[39] Tao Wen. 'Json Iterator - Java'. (2021), https://github.com/json-iterator/java

[40] Brown, Paul and Saloranta, Tatu. 'Jackson'. (2021), https://github.com/FasterXML/jackson

[41] Alibaba. 'FastJson Library'. (2021), https://github.com/alibaba/fastjson

[42] 'Gson'. (2021), https://github.com/google/gson

[43] FAIR. 'OpenCMW C++ Serialiser Benchmarks'. (2021), https://github.com/fair-acc/opencmw-benchmarks-cpp

[44] e. a. Daniel Lemire John Keiser. 'simdjson : Parsing gigabytes of JSON per second'. (2021), https://github.com/simdjson/simdjson

[45] D. Lübke, O. Zimmermann, C. Pautasso, U. Zdun and M. Stocker, 'Interface evolution patterns: Balancing compatibility and extensibility across service life cycles', in *Proceedings of the 24th European Conference on Pattern Languages of Programs*, ser. EuroPLop '19, Irsee, Germany: Association for Computing Machinery, 2019. DOI: 10.1145/3361149.3361164.

[46] A. Jesse. 'API evolution the right way'. (2019), https://opensource.com/article/19/5/api-evolution-right-way

[47] D. Aas. 'Javalin - A simple web framework for Java and Kotlin'. (2021), https://github.com/tipsy/javalin

[48] R. J. Steinhagen and R. Bär, 'FAIR Common Specification: Accelerator and Beam Modes', GSI & FAIR, Tech. Rep., 2017. https://edms.cern.ch/document/1823352/

[49] H. Hüther, J. Fitzek, R. Müller and A. Schaller, 'Realization of a Concept for Scheduling Parallel Beams in the Settings Management System for FAIR', in *Proceedings of ICALEPCS'15*, Melbourne, Australia, 2015.

[50] R. J. Steinhagen *et al.*, 'FAIR Common Specification: Digitization of Analog Signals at FAIR', GSI & FAIR, Tech. Rep., 2017. https://edms.cern.ch/document/1823376/

[51] J. Serrano *et al.*, 'White Rabbit: Sub-nanosecond Timing Distribution over Ethernet', in *Proc. of 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Brescia, Italy: IEEE, Oct. 2009, pp. 1–4. DOI: 10.1109/ISPCS.2009.5340196.

[52] S. Deghaye, M. Lamont, L. Mestre, M. Misiowiec, W. Sliwinski and G. Kruk, 'LHC Software Architecture (LSA) – Evolution toward LHC Beam Commissioning', in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, pp. 307–309.

[53] R. Müller, J. Fitzek and D. Ondreka, 'Evaluating the LHC Software Architecture for Data Supply and Setting Management within the FAIR Control System', in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'09)*, GSI Helmholtzzentrum, Darmstadt, Germany, Kobe, Japan, Oct. 2009, pp. 697–699.

[54] D. Ondreka, J. Fitzek, H. Liebermann and R. Müller, 'Settings Generation for FAIR', in *Proc. of International Particle Accelerator Conference (IPAC'12)*, GSI Helmholtzzentrum, Darmstadt, Germany, New Orleans, Louisiana, USA, May 2012, pp. 3963–3965.

[55] M. Arruat *et al.*, 'Front-End Software Architecture (FESA)', in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, pp. 310–312.

[56] S. Matthies, H. Bräuning, A. Schwinn and S. Deghaye, 'FESA3 Integration in GSI for FAIR', in *Proc. 10th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'14)*, Karlsruhe, Germany, Oct. 2014, pp. 43–45.

[57] V. Rapp and W. Sliwinski, 'Controls Middleware for FAIR', in *Proc. 10th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'14)*, Karlsruhe, Germany, Oct. 2014, pp. 4–6.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

INTEGRATION OF OPC UA AT ELBE

K. Zenker*, R. Steinbrück, M. Kuntzsch,
Institute of Radiation Physics, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany

Abstract

The Electron Linac for beams with high Brilliance and low Emittance (ELBE) at Helmholtz-Zentrum Dresden-Rossendorf (HZDR) is operated using the SCADA system WinCC by Siemens. The majority of ELBE systems is connected to WinCC via industrial Ethernet and proprietary S7 communication. The integration of new subsystems based on MicroTCA.4 hardware, which do not provide S7 communication interfaces, into the existing infrastructure is based on OPC UA using the open source C++ software toolkit ChimeraTK.

This paper discusses OPC UA related contributions to ChimeraTK, that cover an OPC UA backend implementation, development of logging, data acquisition and history modules and improvements of the control system adapter for OPC UA in ChimeraTK. Furthermore, a user data interface based on OPC UA for ELBE is introduced and one ELBE real-life example is described, that makes use of all the afore-mentioned features.

INTRODUCTION

The Electron Linac for beams with high Brilliance and low Emittance (ELBE) at Helmholtz-Zentrum Dresden-Rossendorf (HZDR) is in operation since 2001. It is operated using the SCADA system WinCC by Siemens. The majority of ELBE systems is connected to WinCC [1] via industrial Ethernet and proprietary S7 communication. However, in recent years new subsystems had to be integrated into the existing infrastructure, which do not provide S7 communication interfaces. The Open Platform Communication (OPC) interoperability standard for the secure and reliable exchange of data in industrial automation environments is developed and maintained by the OPC foundation [2]. Its Extension called OPC Unified Architecture (OPC UA) [3] is an open standard for information modeling and machine-to-machine communication, that features e.g. build-in security/authentication and is the foundation of the so-called 4th industrial revolution in industry. It is platform independent and different open source implementations exists that provide support for various development environments like e.g. Python, C/C++, Java and Labview. Because of those features OPC UA has been chosen at ELBE to establish a link between the new subsystems and S7 devices.

As shown in [4] two types of communication were established in the past:

- OPC UA communication between an application and MicroTCA.4 [5] based hardware, i.e. FPGA via direct memory access (DMA) over PICe
- OPC UA communication between an application and Siemens S7-300/400 PLCs

The former communication is established based on the ChimeraTK framework introduced in the following section. The latter communication is based on commercially available OPC UA gateways introduced after. In the following sections OPC UA related components of ChimeraTK and an OPC UA based ELBE machine data interface are presented. Thereafter, the use of the new features is demonstrated by means of an ELBE subsystem as a real-life example.

OPC UA IN CHIMERATK

ChimeraTK [6, 7] is a C++ based open source toolkit for modular control application developments. ChimeraTK provides dedicated libraries for access to devices/data (DeviceAccess library [8]), the application itself (ApplicationCore [9]) and a control system server (ControlsystemAdapter [10]). Both, the Device Access Library and the Control System Adapter library, can be used to implement additional so called device backends and control system adapter implementations. The general structure is shown in Fig. 1. Highlighted in green are parts of ChimeraTK that were contributed as part of the OPC UA developments at ELBE.

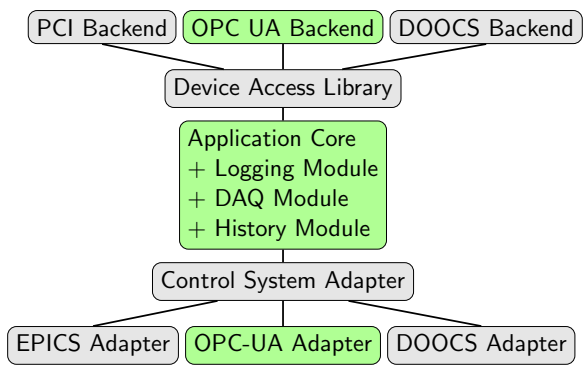


Figure 1: Overview of ChimeraTK. Only selected control system adapters and backends are shown. Contributions discussed in this paper are shown in green.

In general, so far the OPC UA communication in ChimeraTK is implemented using client/server communication, based on TCP/IP. This means no real-time communication can be realized using this approach. Real-time communication would only be possible using the Publish/Subscribe mechanism of OPC UA in combination with Time-Sensitive Networking (TSN) and hardware that supports TSN.

* k.zenker@hzdr.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

Control System Adapter

The control system adapter that allows building OPC UA servers [11] was originally developed in cooperation with the Technical University of Dresden and DESY[12]. This work is presented in [3]. Now it is further developed together with the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB [13], who are also involved in the development of the OPC UA stack open62541 [14], which is used to implement the OPC UA control system adapter in ChimeraTK. Its main task is to expose all process variables and attached meta data like the variable description and engineering unit defined in the ChimeraTK application via an OPC UA server with corresponding nodes. The nodes are only updated with values of the corresponding ChimeraTK application variable upon request. If no client is connected to the OPC UA server updates of the process variables in the ChimeraTK application are not reflected on the OPC UA server side.

A major improvement compared to the version discussed in [3] is the extended mapping feature of the OPC UA control system adapter. In the past it was only possible to add additional static process variables to the variable tree via a mapping file. Now it is possible to extend the variable tree defined in the ChimeraTK application in an arbitrary manner, by copying or linking variables and complete folders including subfolders to different places in the variable tree. In addition, variable names and the meta data can be edited. If the variable name is changed the variable is copied in the variable tree, meaning a new OPC UA node is created. Else if not explicitly stated different in the mapping file the new location is simply linked to the source variable. This is done by adding a reference to the source OPC UA node keeping the number of nodes of the OPC UA server constant. Such a mapping feature is useful in case client applications expect a certain variable structure. For example, when the ChimeraTK application and its variable tree is changed, client applications are still operational if the previous variable tree is created via the mapping. This allows to decouple client developments from server developments.

Another major change is that numeric node identifiers of process variables, which were initially used for all variables of the ChimeraTK application, were replaced by string node identifiers. The string node identifier is created from the variable name and path as created in the ChimeraTK application. This solves the problem that node identifiers changed, when changing the variable tree of the ChimeraTK application. It happened, because node identifiers were assigned dynamically at server start by scanning the applications variable tree. Using a string node identifier guarantees a static node identifier as long as the variable name is not changed in the application.

Furthermore, changes in ChimeraTK, like the introduction of a data validity flag, were reflected in the OPC UA control system adapter by adding corresponding meta data to variables. Only recent changes in ChimeraTK, i.e. the introductions of the new data types *Bool* and *Void*, are not reflected yet.

Finally, the version of the open62541 stack was upgraded from version 0.2 to version 1.2. This new version includes full encryption support. An encryption endpoint for the OPC UA server will be added in the near future, thus two endpoints will be available following the security policies *None* and *Basic256Sha256* as defined in [15]. The *Basic256Sha256* endpoint will support only encrypted and signed messages and thus ensure a high security level.

Application Core

Different control systems supported by ChimeraTK include different control system specific features, e.g. for data archiving, that are not part of ChimeraTK. In case of OPC UA such features could be implemented in the OPC UA control system adapter utilizing the full functionality of OPC UA and e.g. its historizing mechanism. However, it was decided to add features that are needed at ELBE directly to the ApplicationCore library of ChimeraTK. Therefore, they can be used by all control system adapters of ChimeraTK applications.

First, a logging module was added. It covers four different severity levels and three different target streams – a message variable in the control system, a logging file and standard output to the console – that can be combined. The logging level, target stream and the number of messages included in the message variable can be configured at run time.

Second, a module for data acquisition (DAQ) and archiving was developed [16]. It includes two different backends – a ROOT [17] based and HDF5 [18] based backend. The backend type can be defined in the server configuration that is loaded at server start. The HDF5 backend does not yet cover compression and stores any ChimeraTK data type to a float data type in the HDF5 file. In case of the ROOT backend loss-less compression based on the ZStd algorithm [19] is supported and ChimeraTK data types are mapped to native ROOT data types, which improves the data compression rate.

In addition to this module for data archiving a software package with dedicated tools was developed [20]. The main component of this packages is a Graphical User Interface (GUI), that allows DAQ data inspections. It is a python GUI based on the widget toolkit Qt [21] and on the scientific graphics and GUI library pyqtgraph [22]. It can construct time lines out of the DAQ data and allows to visualize specific data periods. Also, it is possible to jump to events that fulfill a specific threshold criteria, e.g. that a certain temperature is above a certain threshold level. A table allows to inspect a statistical summary (mean and standard deviation) for time line data.

Finally, a history module was added, that allows to create a RAM based history for selected process variables. The history is available as a pair of two process variables each holding a data array. One variables holds the variable values and a second variable holds corresponding time stamps. Data of the history module is used in Human Machine In-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

terfaces (HMI), that show actual data trends for ChimeraTK applications. It complements the data available by the DAQ module described above with data that is not yet written by the DAQ module.

Device Access

The device access library allows access to devices that are not necessarily part of the machine, that is running the ChimeraTK application. E.g. as shown in Fig. 1, there is a PCIe backend that allows to access FPGA data via PCIe communication. At ELBE typically the FPGA is part of a MicroTCA.4 platform and it is added by the MicroTCA Carrier Hub (MCH) to the computers PCIe root complex.

In addition to such a classical device access often a communication between different applications is needed. Such a cross application communication can also cover communication between different control systems, e.g. if a ChimeraTK application using the OPC UA Control System Adapter communicates with a DOOCS [23] server via the DOOCS backend shown in Fig. 1. In case of OPC UA, this requires an OPC UA backend [24], which was developed at ELBE. In fact the backend is an OPC UA client and features synchronous and asynchronous data access. The latter is based on subscriptions in combination with items to be monitored that are attached to the subscription. Two methods to select process variables from the ChimeraTK application, i.e. the OPC UA server, to be monitored are available:

1. Map file based process variable selection.
2. Automatic process variable selection.

If using the map file, one needs to specify the OPC UA node identifier, the corresponding OPC UA name space and a name that is used to map the variable to ChimeraTK. The automatic mapping only works if one connects to an OPC UA server that is based on the ChimeraTK OPC UA control system adapter. It maps all process variables available on the server one to one to the ChimeraTK application. In some cases this might be not sufficient to create a process variable tree as needed by the application. In that case, one can use an additional mapping layer provided by ChimeraTK [7], that allows to restructure the variable tree of devices. In addition to the server address and port, the sampling interval can be specified in the backend configuration. It defines the update rate of monitored process variables independent of the update rate of the source process variable of the OPC UA server. The publishing interval is set equal to the sampling interval, which means each sampled value is sent without buffering directly to the backend by the OPC UA server.

OPC UA GATEWAY INFRASTRUCTURE

As discussed in [4] at ELBE commercially available OPC UA gateways [25] (UA Link) by IBHsoftec are used as a data bridge between Siemens S7-300/400 PLCs and OPC UA based applications. Initially only used for the LLRF system introduced in the following section, the gateway infrastructure was developed further to provide an OPC UA based ELBE machine data interface in general. A hierarchy of different gateways as shown in Fig. 2 was set up at ELBE.

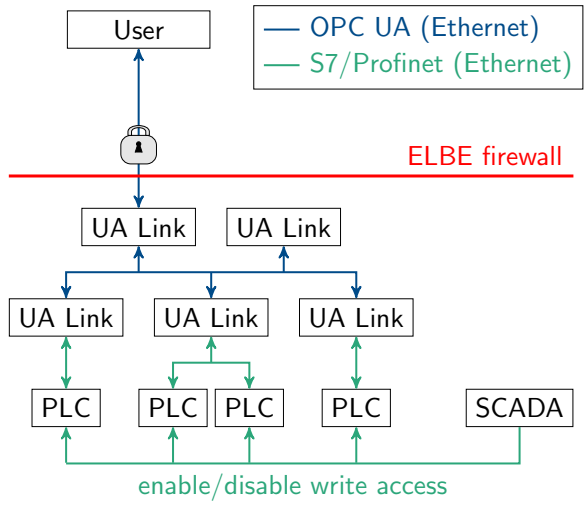


Figure 2: UA Link hierarchy used at ELBE.

In the lowest hierarchy level, UA Link devices are connected to individual PLCs. Already at this level data of different PLCs is aggregated at ELBE. At ELBE typically about 20000 PLC process variables per UA Link are made available via OPC UA. In the hierarchy level above two aggregating UA Links are connected to all the UA Links in the hierarchy level below. They serve as access points to the OPC UA user interface and include an identical process variable tree, that is a a subset of the process variables available in the hierarchy level below. If needed UA Links at this level can also be replaced by workstations running a software[26] that directly uses the configuration of the UA Links.

One of the aggregating UA Link acts as the access point for external users via an encryption based secured connection. For this UA Link machine data is accessible read-only. The second UA Link in this hierarchy level is used for ELBE internal applications. Read and write access is granted to this UA link. However, all PLC registers published via the OPC UA user interface are duplicated on the PLC, which allows to disable write access on the PLC level. Thus, only the duplicates are accessed by the UA Link and operators can control external write access via OPC UA for dedicated process variables or at all from the SCADA system. The duplication also allows data type adjustment and scaling. Restructuring and renaming of process variables is done in the upper UA Link hierarchy level.

In addition to PLC data published via the UA Link, special process variables dedicated to the users can be added to the variable tree on the UA Link. Users can have read and write access to those variables. One example for such variable is a count rate measured by the users and exposed via OPC UA to the SCADA system. Like that such information is available to operators for machine setting optimization.

User authentication for external users is done via user certificates following the X.509 standard in combination with a Certification Authority (CA). A Public-Key-Infrastructure

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

(PKI) including a ROOT CA and dedicated user group CAs was set up at ELBE. Thus, only the CAs certificates and the corresponding revocation lists need to be placed on the UA Link, which simplifies the configuration. User certificates are handed out by the ELBE group and are signed with the corresponding user group CA certificate.

EXAMPLE: THE LLRF SUBSYSTEM AT ELBE

The digital low level radio frequency (LLRF) system of ELBE was the first sub system building on the OPC UA control system adapter discussed here. It is used to control the amplitude and phase of cavities used for particle acceleration and bunching at ELBE [27]. Operators interact with the LLRF system via the SCADA system WinCC. It runs OPC UA clients that are connected to the ChimeraTK LLRF applications (LLRF server). Currently at ELBE seven LLRF servers are running. A simplified overview of the system including only a single LLRF server is shown in Fig. 3.

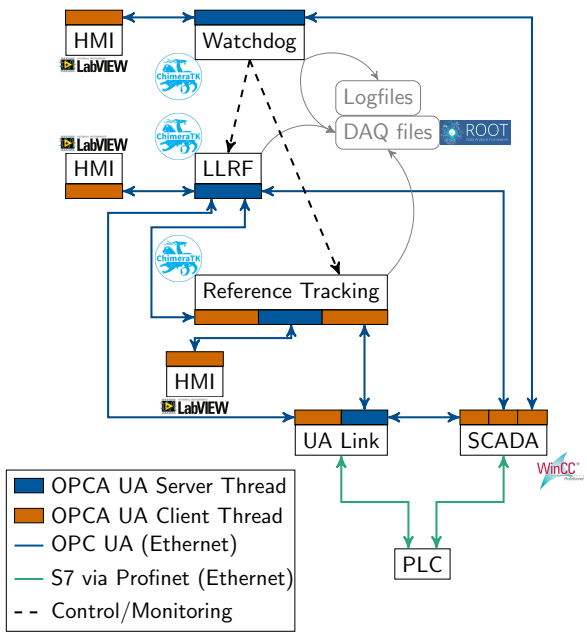


Figure 3: Overview of the control system infrastructure related to the LLRF system at ELBE. Here only one out of seven LLRF applications is shown.

The LLRF servers are controlled and monitored by a second ChimeraTK application called Watchdog server[28], which also monitors the computer where the applications are running. WinCC runs also an OPC UA client that is connected to the Watchdog server, which allows to forward the LLRF Server status to WinCC and also to restart LLRF severs via WinCC. Logging output of all applications controlled by the watchdog are available via OPC UA and it is written in addition to log files stored on disc via the logging module discussed above.

A simple feedback algorithm [27] correcting long term drifts of amplitude and phase introduced by temperature variations is implemented in a separate ChimeraTK application called Reference Tracking server (see also [27]). This server needs data from the LLRF servers as well as machine data, which is read from ELBE PLCs via the UA Link by the OPC UA backend. Calculated corrections are written back to the LLRF server via the OPC UA backend. The Reference Tracking server calculates corrections for all LLRF servers meaning it creates seven OPC UA client threads for the LLRF servers and one OPC UA client thread to the UA Link.

All ChimeraTK applications include the DAQ module introduced before and produce ROOT data files that are written to disc as indicated in Fig. 3. In addition to the WinCC interface to the ChimeraTK servers, application specific expert HMI GUIs have been set up at ELBE. They are implemented using LabVIEW, which features an OPC UA client. Details about LabVIEW client applications are presented in [4].

CONCLUSION

The SCADA system of ELBE is the S7 communication based software WinCC. New subsystems that are integrated to ELBE, like the LLRF system that is based on the MicroTCA.4 platform, do not support S7 communication. In this paper we have shown how such systems are integrated using OPC UA. The integration is based on the C++ toolkit ChimeraTK. The already existing OPC UA control system adapter was extended by additional mapping features that allow to decouple the HMI and application development. Also it was updated to the latest version of the open62541 OPC UA stack it is based on. That allows to make full usage of the encryption provided by OPC UA. In addition ChimeraTK was extended by an OPC UA backend, common modules for logging, data acquisition and online data history, to fulfill the needs at ELBE. Those developments are not only relevant if using the OPC UA control system adapter of ChimeraTK. All other supported control systems benefit, either by the common modules or by the OPC UA backend that allows to integrate data from any OPC UA server to any ChimeraTK application.

The OPC UA based machine data interface presented in this paper, allows external users secure access to ELBE data, which was not available before.

In future, we plan to support more features of OPC UA in the OPC UA control system adapter, like support for OPC UA events or registration to local or global discovery servers according to the OPC UA standard.

ACKNOWLEDGEMENTS

The authors would like to thank our IOSB collaborators and the ChimeraTK team, especially Martin Killenberg and Martin Hierholzer, for their continuous support and feedback.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] *SIMATIC WinCC SCADA*, Siemens. https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada.html

[2] *OPC Unified Architecture*, The OPC Foundation. http://opcfoundation.org/opc-ua/

[3] "OPC Unified Architecture - Part 1: Overview and Concepts", Tech. Rep. IEC TR 62541-1:2020, 2020. https://www.vde-verlag.de/iec-normen/249339/iec-tr-62541-1-2020.html

[4] R. Steinbrück *et al.*, "Control System Integration of a MicroTCA.4 Based Digital LLRF Using the ChimeraTK OPC UA Adapter", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017*, (Barcelona, Spain), Geneva, Switzerland: JACoW Publishing, Jan. 2018, pp. 1811–1814, ISBN: 978-3-95450-193-9.

[5] PICMG, *PICMG specification MTCA.4 Rev. 1.0*, Aug. 2011. https://www.picmg.org/openstandards/microtca

[6] M. Killenberg *et al.*, "Abstracted Hardware and Middleware Access in Control Applications", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17)*, (Barcelona, Spain), Geneva, Switzerland: JACoW Publishing, Jan. 2018, pp. 840–845, ISBN: 978-3-95450-193-9. DOI: 10.18429/JACoW-ICALEPCS2017-TUPHA178.

[7] G. Varghese *et al.*, "ChimeraTK - A Software Tool Kit for Control Applications", in *Proc. of International Particle Accelerator Conference (IPAC'17)*, (Copenhagen, Denmark), ser. International Particle Accelerator Conference, Geneva, Switzerland: JACoW, May 2017, pp. 1798–1801, ISBN: 978-3-95450-182-3.

[8] *Device Access library*, ChimeraTK. https://github.com/ChimeraTK/DeviceAccess

[9] *ApplicationCore library*, ChimeraTK. https://github.com/ChimeraTK/ApplicationCore

[10] *ApplicationCore library*, ChimeraTK. https://github.com/ChimeraTK/ControlSystemAdapter

[11] *OPC UA Control System Adapter*, ChimeraTK. https://github.com/ChimeraTK/ControlSystemAdapter-OPC-UA-Adapter

[12] Deutsches Elektronen-Synchrotron. https://www.desy.de

[13] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB. https://www.iosb.fraunhofer.de

[14] *open62541 — An open source and free C (C99) OPC UA stack licensed under LGPL + static linking exception*. http://open62541.org

[15] "OPC Unified Architecture - Part 7: Profiles", Tech. Rep. IEC 62541-7:2020 RVL, 2020. https://www.vde-verlag.de/iec-normen/248879/iec-62541-7-2020-rlv.html

[16] K. Zenker, *DAQ Module*. https://github.com/ChimeraTK/ApplicationCore-MicroDAQ

[17] R. Brun and F. Rademakers, "Root — an object oriented data analysis framework", *Nucl. Inst. & Meth. in Phys. Res. A*, vol. 389, no. 1, pp. 81–86, 1997, See also "ROOT" [software], Release v6.20/04, 01/04/2020. DOI: 10.5281/zenodo.3895855.

[18] The HDF Group, *Hierarchical data format version 5*. http://www.hdfgroup.org/solutions/hdf5/

[19] Y. Collet and E. M. Kucherawy, "Zstandard compression and the application/zstd media type", *RFC 8478*, 2018. DOI: 10.17487/RFC8478.

[20] K. Zenker, *DAQ Tools*. https://github.com/ChimeraTK/ApplicationCore-MicroDAQ-Tools

[21] *GUI widget tool kit Qt*, The Qt Company. https://code.qt.io/cgit/qt/qtbase.git

[22] *PyQtGraph - Scientific Graphics and GUI Library for Python*. https://www.pyqtgraph.org

[23] K. Rehlich *et al.*, "DOOCS: an Object Oriented Control System as the Integrating Part for the TTF Linac", in *Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'97)*, (Beijing, China), 1997. https://www3.aps.anl.gov/News/Conferences/1997/icalepcs/schedual.html

[24] *OPC UA Device Access*, ChimeraTK. https://github.com/ChimeraTK/DeviceAccess-OpcUaBackend

[25] *IBH Link UA*, IBHsoftec. https://www.ibhsoftec.com

[26] *IBH OPC UA Server/Client*, IBHsoftec. https://www.ibhsoftec.com

[27] K. Zenker *et al.*, "Microtca.4-based low-level rf for continuous wave mode operation at the elbe accelerator", *IEEE Transactions on Nuclear Science*, vol. 68, no. 9, pp. 2326–2333, 2021. DOI: 10.1109/TNS.2021.3096757.

[28] K. Zenker, *Watchdog application*. https://github.com/ChimeraTK/Watchdog

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV011

INTERFACING EPICS AND LabVIEW USING OPC UA FOR SLOW CONTROL SYSTEMS

Jalal Mostafa, Armen Beglarian, Suren A. Chilingaryan, Andreas Kopmann
Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany

Abstract

The ability of EPICS-based control systems to adapt to heterogeneous architectures made EPICS the defacto control system for scientific experiments. Several approaches have been made to adapt EPICS to LabVIEW-based cRIO hardware, but these approaches, including NI EPICS ServerI/O Server: (1) require a lot of effort to maintain and run, especially if the controllers and the process variables are numerous; (2) only provide a limited set of metadata; or (3) provide a limited set of EPICS features and capabilities. In this paper, we survey different solutions to interface EPICSwith LabVIEW-based hardware then propose EPICS OPCUA device support as an out of the box interface between LabVIEW-based hardware and EPICS to preserve most of EPICS features and provide reasonable performance for slow control systems.

INTRODUCTION

The KArlsruhe TRItium Neutrino (KATRIN) experiment is a large-scale scientific experiment to determine neutrino mass using Tritium beta decay [1]. Large-scale scientific experiments like KATRIN employ diverse hardware from different vendors to support the control system infrastructure, e.g., NI cRIO, Siemens S7, custom hardware chips, etc. For instance, KATRIN employs around 10,000 process variables that are distributed in bunches of 100 to 300 process variables on different NI cRIO, NI cFieldPoint, and Siemens S7 devices. The heterogeneity of such systems imposes a new challenge of providing a unified layer of control and interoperability between these heterogeneous systems and other services like alerts, data archiving, and analysis. The Experimental Physics and Industrial Control System (EPICS) solves this problem using an intermediate C++ software layer.

EPICS is a set of tools and libraries to develop distributed server-client control systems for large-scale scientific experiments e.g. particle accelerators. It can solve the challenge of heterogeneity by acting as an abstract software layer for all devices through implementing a C++ abstraction interface called Device Support. Operator's setpoints and sensor measurements are then channeled through EPICS server or Input/ Output Controller (IOC) using one of EPICS network protocols: Channel Access (CA) and its successor Process Variable Access (PVA). Figure 1 shows a simplified illustration of the EPICS server architecture of the EPICS server. An EPICS server keeps all process variables in in-memory structures called Database. When a client asks for a process variable, the EPICS server access these structures using a module called Database Access. The EPICS server can read data from the hardware using the device support layer where the data is mapped to the corresponding process variable through database access and then published on the network using CA or PVA. Operator's setpoints work similarly, but in the opposite direction: an EPICS client (the operator) sets a process variable on an EPICS server using CA or PVA, which is implemented on the hardware using Device Support.

This architecture imposes new challenges on the interoperability between EPICS and NI LabVIEW-based cRIOs. The LabVIEW programming environment offers very tight integration with EPICS. Several approaches have been made to provide an integration for LabVIEW with EPICS, but they usually require an effort to maintain and run, provide a limited set of metadata, or provide a limited set of EPICS features and capabilities. In this paper, we propose an EPICS-LabVIEW integration based on Open Platform Communications Unified Architecture (OPC UA) protocol.

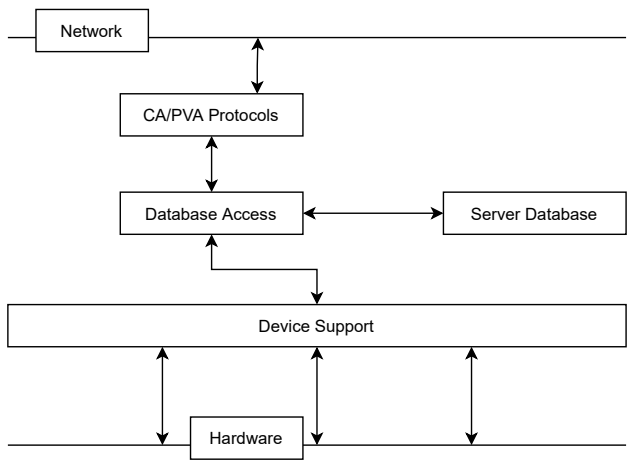


Figure 1: EPICS Server Simplified Internal Architecture.

RELATED WORKS

Several works have been done before to interface LabVIEW with EPICS: most notable is *EPICS Server I/O Server* [2]. *EPICS Server I/O Server* is a complete implementation of EPICS CA protocol for LabVIEW from NI. It allows the developer to create LabVIEW Shared Variables to publish EPICS process variables on the network, a complicated process that involves many clicks to accomplish a simple task, especially when the process variables are numerous. The metadata that EPICS Server I/O Server provides is limited to alarms only leaving behind important metadata like description and engineering units.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Another LabVIEW/EPICS interface that depends on Shared Variables is *NetShrVar* [3]. NetShrVar uses the proprietary NI-PSP protocol to bind a process variable in an EPICS C++ server to a LabVIEW shared variables using a set of complex configurations written in XML.

lvPortDriver [4] allows users to develop EPICS device support in LabVIEW code and to start an EPICS C++ server through the same LabVIEW program. This is possible through an *asynDriver* [5] C++ layer and a LabVIEW VI library. Although *lvPortDriver* inherits platform independency from EPICS but compilation and deployment are still needed for each specific NI target CPU architecture.

[6–8] employs shared memory between LabVIEW program and EPICS C++ server to interface EPICS with LabVIEW. Other than being in need to compile for each NI target architecture, these solutions are platform-specific and they are hard to port from one operating system to another.

IRIO [9], *Nheengatu* [10] allow developers to run EPICS on cRIO and other NI devices based on a middleware library that allows direct access to NI hardware, but it depends on the target architecture.

EPICS-LABVIEW INTERFACE USING OPC UA

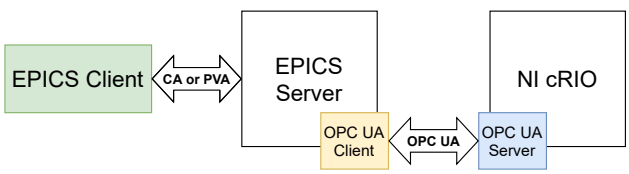


Figure 2: EPICS/OPC UA Architecture.

OPC UA provides seamless integration with EPICS. It is an open service-oriented protocol that can run on any hardware and operating system and thus provides a good fit for interoperability between EPICS and LabVIEW-based hardware.

Figure 2 shows how EPICS can interface with LabVIEW. The LabVIEW developer writes standard OPC UA code on the target device. An standard EPICS server can communicate to the OPC UA server running on the LabVIEW target using a device support module for OPC UA [11].

The OPC UA Device Support for EPICS provides the easy representation of hierarchical data structures, access control, and data monitoring. All standard data types for both input and output are supported. It adds an extra record type *opcuaItem* which allows representing OPC UA objects in EPICS by destructing the OPC UA object to standard data types records.

The OPC UA server/client architecture abstracts the communication of EPICS with the cRIO and thus allows deploying a one-time compilation EPICS server on any commodity server or virtual machines or containers to add modularity, isolation, and scalability to the architecture.

PERFORMANCE EVALUATION

To evaluate this architecture for slow control systems like KATRIN, we launch an OPC UA server on NI cRIO 9047 and then run an EPICS server running on a workstation with Intel Xeon CPU E5-1630 v3 @ 3.70 GHz and 1 Gbps Ethernet connection, and linked to the cRIO through the EPICS OPC UA Device Support module. The OPC UA objects in our experiments are represented using a flat data architecture: each process variable is represented by an OPC UA variable inside a distinct OPC UA object. We operate experiments by simulating sensors (read from cRIO to EPICS) and setpoints (write to cRIO through EPICS). Table 1 shows the experimental setup of 300 process variables based on the requirements of the KATRIN control system.

Table 1: Update Rate and Expected № of Elements for Each of the 300 Values

Update Rate	Exp. № of Values in 1 Hr.
10 Hz	36000
2 Hz	7200

Sensor Mode

We launch two experiments for the sensor mode evaluation where, in both experiments, the OPC UA server on the cRIO is generating incremental values for each of the 300 process variables for a duration of an hour: (1) 2 Hz experiment, one value for each of the process variables every 500 ms; and (2) 10 Hz experiment, one value every 100 ms. The EPICS server is monitoring all the 300 variables in both experiments over one OPC UA subscription, then we use *PyEPICS*[12] to read the values from the EPICS variables through CA monitoring. The EPICS OPC UA Device Support is configured to use the timestamp from the OPC UA protocol.

In an hour interval, we noticed that the expected number of values for each record is attained. No loss of data in both 10 Hz and 2 Hz experiments.

Setpoints Mode

We simulate setting values to actuators through EPICS and that are controlled by the OPC UA server. We then launch an EPICS client developed in C with real-time priority threads (one thread per process variable), as illustrated in Fig. 3. We want to check two metrics in this mode: lost values rate and response time (time delay between setting the value in EPICS and implementing the setpoint in the OPC UA server). This is possible through having two records for each EPICS process variable in the EPICS database: (1) the *ao* output record to output the variable to the OPC UA server using EPICS timestamp, and (2) the *ai* readback record to read the process variable values back using an OPC UA subscription and OPC UA timestamp. We then monitor both records; the loss rate is then computed by comparing values from both monitors (output and readback).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV011

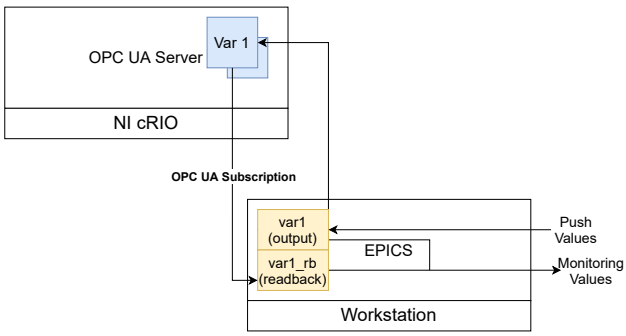


Figure 3: Setpoints Mode Experimentation.

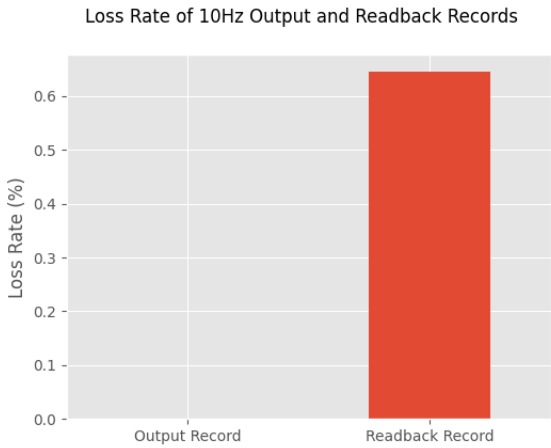


Figure 4: Comparison of Data Loss between the Output Record and the Readback Record in 10 Hz Experiment for the 300 Process Variables.

While the loss at 2 Hz is zero in the output and the readback records, the loss rate at 10 Hz is zero for the output record but around 0.6% for the readback in an hour experiment interval which means EPICS has processed the setpoints very well, but the OPC UA server did not implement them. Figure 4 shows the loss rate of the output record using the output monitor and that of the readback record using the readback monitor.

CONCLUSION

Using OPC UA protocol can be a good connection interface to integrate EPICS C++ code and NI LabVIEW-based hardware without a strong impact on performance for slow control systems. Comparing both 2 Hz and 10 Hz experiments, we notice the performance is limited by the OPC UA server which depends on the performance of the cRIO or the system hosting it. Therefore, we plan to use this architecture for the KATRIN experiment where the maximum update rate is 2 Hz and can guarantee no data loss.

REFERENCES

[1] J. Wolf, "The KATRIN neutrino mass experiment," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 623, no. 1, pp. 442–444, 2010, 1st International Conference on Technology and Instrumentation in Particle Physics, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2010.03.030. https://www.sciencedirect.com/science/article/pii/S0168900210005942

[2] A. Veeramani, T. Debelle, W. Blokland, R. Dickson, A. Zhukov, and O. RAD, "Options for interfacing EPICS to COTS hardware through LabVIEW," 2009, THD004.

[3] Network shared variable epics support module, (2013), http://epics.isis.stfc.ac.uk/doxygen/NetShrVar/

[4] lvPortDriver, (2020), https://github.com/lanl/lvPortDriver

[5] Epics asyn driver, (2021), https://epics.anl.gov/modules/soft/asyn/

[6] D. Thompson and W. Blokland, "A shared memory interface between LabVIEW and EPICS," in *Proceedings of ICALEPCS2003*, Korea, 2003, pp. 275–277.

[7] A. Liyu, W. Blokland, and D. Thompson, "Labview library to epics channel access," in *Proceedings of the 2005 Particle Accelerator Conference*, 2005, pp. 3233–3234. DOI: 10.1109/PAC.2005.1591423.

[8] G. Li and J. Zhao, "Application of LabVIEW-EPICS in measuring and monitoring system of BEPCII," *Nuclear Electronics and Detection Technology*, vol. 26, no. 2, pp. 222–225, 2006.

[9] M. Ruiz *et al.*, "IRIO technology: Developing applications for advanced DAQ systems using FPGAs," in *2016 IEEE-NPSS Real Time Conference (RT)*, 2016, pp. 1–5. DOI: 10.1109/RTC.2016.7543090.

[10] D. Alnajjar, G. Fedel, and J. Piton, "Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems*, Oct. 2019. DOI: 10.18429/JACoW-ICALEPCS2019-WEMPL002.

[11] R. Lange *et al.*, "Integrating OPC UA Devices in EPICS," in *18th International Conference on Accelerator and Large Experimental Physics Control Systems*, Oct. 2021, MOPV026.

[12] Pyepics, (2014), https://pyepics.github.io/pyepics/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV012

AUTOMATED DEVICE ERROR HANDLING IN CONTROL APPLICATIONS

M. Killenberg*, J. Georg, M. Hierholzer, C. Kampmeyer[1], T. Kozak, D. Rothe,
N. Shehzad, J. H. K. Timm, G. Varghese[2], C. Willner,
Deutsches Elektronen-Synchrotron DESY, Notkestr. 85, 22607 Hamburg, Germany
[1]now with Sokratel Kommunikations- und Datensysteme GmbH,
Langenhorner Chaussee 625, 22419 Hamburg, Germany
[2]now with European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany

Abstract

When integrating devices into a control system, the device applications usually contain a large fraction of error handling code. Many of these errors are run time errors which occur when communicating with the hardware, and usually have similar handling strategies. Therefore we extended ChimeraTK, a software toolkit for the development of control applications in various control system frameworks, such that the repetition of error handling code in each application can be avoided. ChimeraTK now also features automatic error reporting, recovery from device errors, and proper device initialisation after malfunctioning and at application start.

GOALS AND REQUIREMENTS

If the business logic of a control application is intertwined with code for device opening and communication error handling, it becomes hard to read. Hence, this should be avoided. In an ideal case, the application programmer does not have to write any device handling code, and a framework takes care of all the necessary actions, reports faults to the control system and handles the device recovery. For this to work with many different kinds of devices, a sufficient level of abstraction is necessary.

- The framework has to provide a common API for all devices.
- The framework has to guarantee that the user code can always read and write all its variables, and therefore needs to separate read/write operations in the user code from the actual hardware access.

These are the two key places where the framework has to have an appropriate interface. The second point is a consequence of the fact that the framework is taking care of connecting to a device, initialising the device, reporting the connection status to the control system and propagating information about faulty connections.

THE ChimeraTK FRAMEWORK

ChimeraTK, the ***C**ontrol system and **H**ardware **I**nterface with **M**apped and **E**xtensible **R**egister-based device **A**bstraction **T**ool **K**it*, is a framework for writing control applications. [1] An overview is shown in Fig. 1. ChimeraTK consists of three main components:

- The DeviceAccess library provides a common interface to different device types by introducing a backend plugin mechanism.
- The ControlSystemAdapter allows to integrate into various control system middlewares as a native application. [2]
- The ApplicationCore library connects many application modules, which make up the business logic, with the devices and the control system.

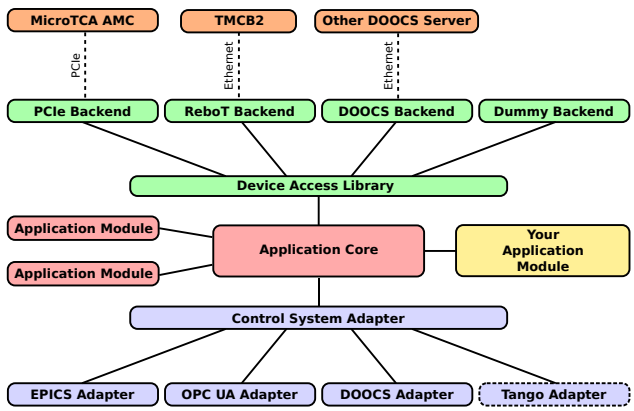


Figure 1: Overview of the ChimeraTK framework. All components connected with solid lines are part of the same executable.

All these components are part of the same executable. The C++-code written by the application programmer consists of application modules for ApplicationCore. ApplicationCore is interfacing to DeviceAccess and the ControlSystemAdapter internally. The user never directly interacts with these libraries on the C++ level. Which devices to use and the parameters for the control system integration are loaded from configuration files at application start.

The DeviceAccess Library

The DeviceAccess library is the interface to various devices. The device usually is a hardware component which is being integrated into the control system, but it can also

* martin.killenberg@desy.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV012

be another control system application. DeviceAccess comes with an extensible backend interface. Support for PCIexpress communication and a suite of dummies for unit testing are built into the base library. Backends for DOOCS [3,4], OPC UA [5,6] and Modbus [7,8] are available as runtime-loadable plugins. A backend for EPICS [9] is currently under construction. [10] The plugin mechanism allows to easily add support for further devices and protocols.

An important abstraction step in DeviceAccess is the introduction of so called register accessors. These are objects representing a device register in the application. It contains a buffer with the data content, and read() and write() functions to synchronise the buffer and the device. Each of those registers is identified by a name. For numerically addressed protocols like PCIexpress or Modbus, a register name mapping is built into the library.

The ApplicationCore Library

The main idea of ApplicationCore is to build the application from small, self contained application modules (see Fig. 2). Each module has a set of input and output variables and its own processing thread. All inputs and outputs are identified by a name, which associates it with a process variable that is part of the variable household of the application. ApplicationCore automatically connects the inputs and outputs of all components (application modules, device modules and the control system module) which access the same process variable. The code inside a module does not care where the data is coming from and where it is going to. Each module is completely self contained.

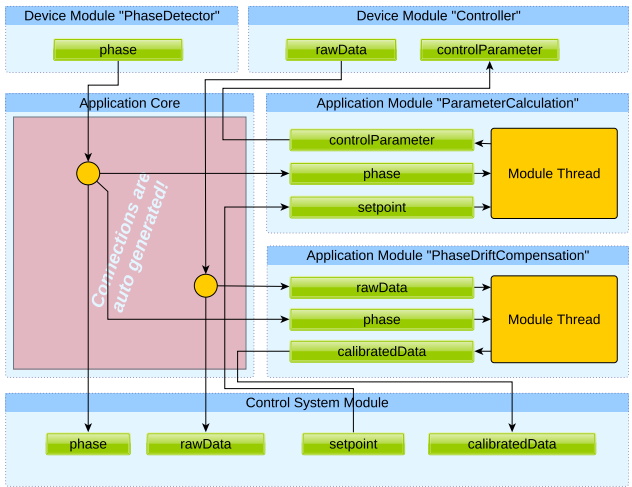


Figure 2: Applications written with ApplicationCore consist of self-contained application modules. Each module has its own thread and interfaces to the rest of the application via input and output variables. The connections of those variables to the device modules and the control system module are automatically generated by ApplicationCore.

Under the hood the variables of the different modules are connected via lock-free queues, which makes ApplicationCore a multi-threading library with very efficient, modern inter-thread communication. However, this is completely transparent to the user as there are no locks involved.

In ApplicationCore there are two kinds of inputs: Push-type inputs and poll-type inputs. Read operations of poll-type inputs return immediately and the variable contains the latest value. Read operations of push-type inputs block if no data is written. Once data is available, the reading thread is woken up. Typically each application module has as least one push-type input which the module thread is waiting for. Like this, the modules automatically synchronise with each other, without the need for locks or sleeps.

The ControlSystemAdapter Library

While the DeviceAccess library is a client interface to the devices, the ControlSystemAdapter turns the application into a server which introduces the published process variables to the control system. It consists of the ControlSystemAdapter base library and a concrete implementation for a control system middleware. ApplicationCore is using the interface of the base library and is completely independent of the actual adapter implementation. The adapter implementation is linked as separate library.

For a proper integration into the control system of a facility, each adapter implementation comes with a mapping layer which allows to change the names of variables to match the facility's naming scheme or create special data types only available for the particular middleware protocol.

Currently ControlSystemAdapter implementations are available for DOOCS [11], EPICS 3 [12, 13] and OPC UA [14].

EXCEPTION HANDLING AND DEVICE INITIALISATION

Handling Faults

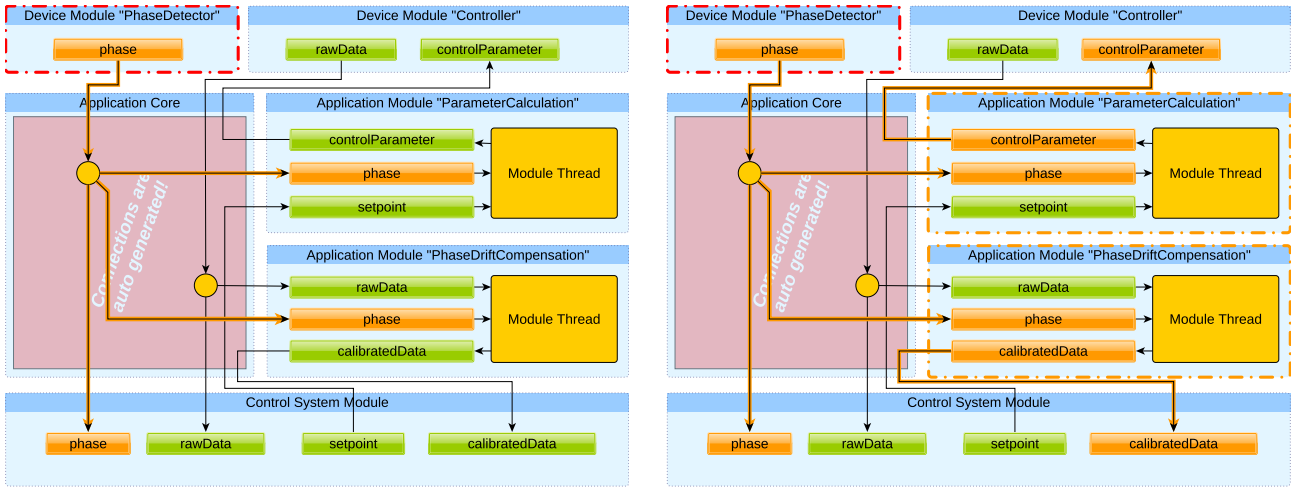
Once a problem, usually a communication error, has been detected in DeviceAccess, this is internally reported to the backend. The backend informs all register accessors about the fault. From this point on, all read() operations to poll type accessors and all write() operations will throw a `runtime_error` exception when being called. All push-type accessors will send exactly one `runtime_error` exception through the queue (instead of data), and then no further data is sent until the next value has been received after device recovery (see discussion in the ApplicationCore section).

Each device backend implementation is responsible for re-establishing the connection when its open() function is called again. Usually this means trying to re-connect via whatever protocol is used to communicate with the device. Like this, no special, device-dependent code is needed on the higher levels.

In ApplicationCore, the exceptions from the accessors are caught and handled. As the main point of the exception handling scheme is to keep the user code free from device handling code, the user code in the application modules must

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV012



(a) The device module "PhaseDetector" has seen a runtime error. All of its outputs are flagged with data validity `faulty` (orange).

(b) All application modules which have at least one input flagged as `faulty` automatically switch the data validity of all their outputs to `faulty`.

Figure 3: Propagation of the data validity flag in ApplicationCore.

not see these exception. The exception is reported to the device module, which in the background is trying to recover by periodically calling open() on the backend. At the same time the device status reported to the control system changes to *not functional*.

Without further actions, the application modules with the user code would not be aware that the device has a problem and that data is not updating any more. For instance if there are poll-type inputs for calibration parameters, which are read in addition to the push-type main data, the application module would keep processing data without noticing that the calibration parameter might be outdated. That's why a mechanism has been introduced to automatically propagate a data validity flag in ApplicationCore.

Whenever an exception is caught in a process variable, the last good data value is re-sent with the data validity set to `faulty` (Fig. 3a). For push-type data this means, that the receiving module will be triggered exactly once. For poll-type data, the same value is read over and over again, but being flagged as `faulty`.

Application modules are small, self contained code blocks where all outputs logically depend on all inputs. This allows for an automatic propagation of the data validity flag: As soon as one input is flagged as `faulty`, all outputs of the module are also flagged as `faulty`. Like this, the flag is propagated for all data that is calculated from `faulty` inputs (Fig. 3b). Notice that in Fig. 3b the `rawData` from the "Controller" device is not marked as faulty, although one of the inputs of the device (`controlParameter`) is invalid. For device modules (and also the control system module) there is no automatic data validity propagation because they are not small, self contained units where all outputs logically depend on all inputs.

This mechanism covers the two possible scenarios: Modules which use the faulty data as their main data, which means they read them as push-type input to synchronise their execution to it, are processing the data exactly once to propagate the data validity flag. Modules which use the faulty data as poll inputs keep running, but flag their data as invalid. A typical use case is the `phase` parameter in Fig. 3. The `phase` is a typically slowly drifting parameter which is used to calibrate the `rawData`. It is a safe assumption that the calibration is still approximately correct, and the facility can keep running with it, even if there is a short communication problem with the phase detector. The calculated `calibratedData` is flagged as `faulty` to indicate that there is a problem. If the problem persists, the calibration will become less and less reliable. The decision whether operation can continue with `faulty` inputs obviously is a case by case decision. Although usually the code in application modules does not care about the data validity because it is propagated automatically, the flag for each input is available, such that critical decisions that depend on data being valid can be done.

Device Recovery

When a device can be re-opened successfully, the application and the device have to synchronise their states. The device might have re-booted, in which case many devices need to be initialised (reset-flags cleared, clocks set, dynamic ranges of ADCs and DACs adjusted etc.). For this to be automated, ApplicationCore provides the possibility to register a callback function. It is provided by the user code and executed immediately after the communication to the device is established. The parameters written in the initialisation phase often are implementation details of the firmware which should not be connected to the control system module. In this function the user code has direct access to the device and all of its registers, without adding process variables to the variable household of the application (in contrast to the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV012

input and output variables of application modules, which are all visible in the control system due to the automatic connection code).

After the device initialisation function has been called, all output variables from the application to the device are written. This assures that the device and the application both have the same parameters everywhere. The device module remembers the last written value of each variable, so it can be restored in the device even if an output is not re-written by the application. This is especially important for parameters which are directly connected from the control system module to the device module. Obviously a user does not want to re-enter all parameters again in a panel after a device has come back online.

In a third step, all push-type data is fetched from the device and written to the application once. This data does not carry the data validity flag `faulty` any more, so the flag is cleared automatically in each module once all of its inputs are back to data validity `OK`.

Finally, after the device is initialised and all data to and from the device has been transferred once, the device status flag, which is reported to the control system, is set to `OK` as well. The application is back to normal operation.

Application Start

At application start, all devices start as *not functional*, and the device recovery procedure is launched. In the section about automatic fault handling it was described that an application module keeps running, even if a poll-type input is flagged with data validity `faulty`. This is working under the assumption that the last good value can be used to continue operation, even if the value is not being updated any more. At the first start of the application, when no communication to the device is established yet, this mechanism is not working because there is no previous good value.

To solve this issue, the concept of an *initial value* has been introduced in ApplicationCore. All application modules require that this first, valid value has been received by all inputs before operation is started. This is assured by ApplicationCore, which only starts the user-written main loop once all initial values are present. Each application module starts by performing its calculation with these initial values, and writing its outputs before doing the first (blocking) read on any input. Subsequent modules now also get their initial values, and the application takes up operation in the correct order. The control system adapter is getting the initial values from the persistency layer of the control system middleware, and is sending them once at application start.

If a device is not available at application start, this means that parts of the application will not start running. This does not pose a problem, it rather is a feature. Only those modules which cannot calculate valid data anyway are waiting, and once these data are available, the sequence of propagating them is automatically started. Even if all the application modules would be waiting for initial values, this does not mean that the application is non-responsive. The communication on the control system side is always working, and the status which device is faulty is shown there. There is nothing more the application can do in this situation, except for waiting for the device to become available.

The start sequence for opening a device is the same as for the recovery case: Once the communication is working, the initialisation callback function is executed, then all process variables from the application are written to the device, and finally the initial values are read for all push-type variables and propagated.

Although it should be avoided, it sometimes is inevitable to have circular dependencies between application modules, which would mean that modules are mutually waiting for the initial value from the other module before they can send their outputs. This circle needs to be resolved manually. One of the modules has to send a safe start value on its output before receiving the initial values. This cannot be done automatically because only a programmer who knows the application logic can decide where to break the circle and which "safe" initial value to send. As unintended circular dependencies prevent a proper application start and are difficult to debug, ApplicationCore comes with a built-in circular dependency detection. If an application module does not receive an initial value until a timeout, it will print which variables are affected to simplify the debugging process.

CONCLUSION

The ChimeraTK framework for the development of control system applications features the ApplicationCore library for the development of modular applications, which interacts seamlessly with the DeviceAccess library for hardware access and the ControlSystemAdapter for the integration into different control system environments. ChimeraTK has lately been extended with automatic device initialisation and exception handling. Special emphasis has been put in a clean initialisation and recovery procedure.

ApplicationCore based device applications are used to operate the low level radio frequency systems at several accelerators using different control system software (EuropanXFEL [15] and FLASH [16] at DESY, Hamburg, using DOOCS; ELBE [17] at HZDR, Dresden, using OPC UA [18]; TARLA [19] in Ankara using EPICS3 and several others). The improved device handling and automatic re-initialisation has reduced the time and manual steps which are required bring the system back online after a major fault or maintenance period. This results in reduced downtimes and a higher availability and reliability of the whole facility.

The ChimeraTK suite is open source software which is published under the GNU General Public License or the GNU Lesser General Public License (depending on the software component). It is available under [20].

REFERENCES

[1] M. Killenberg *et. al.*, "Abstracted Hardware and Middleware Access in Control Applications", in *Proc. ICALEPCS2017*,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Barcelona, Spain, 2017, paper TUPHA178. doi:10.18429/JACoW-ICALEPCS2017-TUPHA178

[2] M. Killenberg *et al.*, "Integrating control applications into different control systems", in *Proc. ICALEPCS2015*, Melbourne, Australia, 2015, paper TUD3O05. doi:10.18429/JACoW-ICALEPCS2015-TUD3O05

[3] The Distributed Object Oriented Control System (DOOCS), http://doocs.desy.de/.

[4] DeviceAccess-DoocsBackend: DOOCS client for the ChimeraTK DeviceAccess library, https://github.com/ChimeraTK/DeviceAccess-DoocsBackend

[5] OPC Unified Architecture Specifications - Part 1: Overview and Concepts, https://reference.opcfoundation.org/v104/Core/docs/Part1/.

[6] DeviceAccess-OpcUaBackend: The OPC UA backend for DeviceAccess, https://github.com/ChimeraTK/OPC_UA_backend

[7] The Modbus Organisation, https://modbus.org/.

[8] DeviceAccess-ModbusBackend: Client supporting tcp and rtu communication with modbus devices, https://github.com/ChimeraTK/DeviceAccess-ModbusBackend

[9] Experimental Physics and Industrial Control System (EPICS), http://www.aps.anl.gov/epics/index.php

[10] DeviceAccess-EPICS-Backend: EPICS client backend for DeviceAccess, https://github.com/ChimeraTK/DeviceAccess-EpicsBackend

[11] ControlSystemAdapter-DoocsAdapter: The DOOCS implementation for the ControlSystemAdapter, https://github.com/ChimeraTK/ControlSystemAdapter-DoocsAdapter

[12] ChimeraTK-ControlSystemAdapter-EPICS: Device support for integrating ChimeraTK-based devices into EPICS-based control-systems, https://github.com/aquenos/ChimeraTK-ControlSystemAdapter-EPICS

[13] ControlSystemAdapter-EPICS-IOC-Adapter: Implementation for the ControlSystemAdapter to create an EPICS IOC, https://github.com/ChimeraTK/ControlSystemAdapter-EPICS-IOC-Adapter

[14] ControlSystemAdapter-OPC-UA-Adapter: The OPC UA implementation for the ControlSystemAdapter, https://github.com/ChimeraTK/ControlSystemAdapter-OPC-UA-Adapter

[15] M. Altarelli *et al.*, "XFEL: The European X-Ray Free-Electron Laser", DESY, Hamburg, Rep. DESY-2006-097, 2007. doi:10.3204/DESY_06-097

[16] C. Schmidt *et al.*, "Real time control of RF fields using a MicroTCA.4 based LLRF system at FLASH", in *19th IEEE Real-Time Conference*, Nara, Japan, 2014. doi:10.1109/RTC.2014.7097430

[17] F. Gabriel *et. al.*, "The Rossendorf radiation source ELBE and its FEL projects", *Nucl. Instr. Meth. B*, vol. 1143, pp. 161-163, 2000. doi:10.1016/S0168-583X(99)00909-X

[18] R. Steinbrück *et. al.*, "Control System Integration of a µTCA.4 based digital LLRF using the ChimeraTK OPC UA Adapter", in *Proc. ICALEPCS2017*, Barcelona, Spain, 2017, paper THPHA166. doi:10.18429/JACoW-ICALEPCS2017-THPHA166

[19] A. Aksoy *et al.*, "TARLA: The First Facility of Tukrish Acceleretor Center (TAC)", in Proc. IPAC2017, Copenhagen, Denmark, 2017, paper WEPAB087. doi:10.18429/JACoW-IPAC2019-THPRB115

[20] ChimeraTK: Control system and Hardware Interface with Mapped and Extensible Register-based device Abstraction Tool Kit, https://github.com/ChimeraTK/.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV013

BACKEND EVENT BUILDER SOFTWARE DESIGN FOR INO mini-ICAL SYSTEM

Mahesh Punna[1], Narshima Ayyagiri[1], Janhavi Avadhoot Deshpande[1], Preetha Nair[1], Padmini Sridharan[1],Shikha Srivastava[1], Satyanarayana Bheesette[2], Yuvaraj Elangovan[2], Gobinda Majumder[2], Nagaraj Panyam[2]

[1]BARC, Mumbai, India
[2] TIFR, Mumbai, India

Abstract

The Indian-based Neutrino Observatory collaboration has proposed to build a 50 KT magnetized Iron Calorimeter (ICAL) detector to study atmospheric neutrinos. The paper describes the design of back-end event builder for Mini-ICAL, which is a first prototype version of ICAL and consists of 20Resistive Plate Chamber (RPC) detectors. The RPCs push the event and monitoring data using a multi-tier network technology to the event builder which carries out event building, event track display, data quality monitoring and data archival functions. The software has been designed for high performance and scalability[chronous data acquisition and lockless concurrent data structures. Data storage mechanisms like ROOT, Berkeley DB, Binary and Protocol Buffers were studied for performance and suitability. Server data push module designed using publish-subscribe pattern allowed transport & remote client implementation technology agnostic. Event Builder has been deployed at mini-ICAL with a throughput of 3MBps. Since the software modules have been designed for scalability, they can be easily adapted for the next prototype E-ICAL with 320 RPCs to have sustained data rate of 200MBps.

INTRODUCTION

The Indian-based Neutrino Observatory (INO) collaboration has proposed to build a 50 KT magnetized Iron Calorimeter (ICAL) detector to study atmospheric neutrinos and to make precision measurements of the neutrino oscillation parameters. The detector will look for muon neutrino induced charged current interactions using magnetized iron as the target mass and around 28,800 Resistive Plate Chambers (RPCs) as sensitive detector elements [1]. The mini-Iron Calorimeter (mini-ICAL) detector, a prototype of the ICAL detector is being set up at the Inter Institutional Centre for High Energy Physics' (IICHEP) transit campus at Madurai. The mini-ICAL detector has 20 glass Resistive Plate Chamber (RPC), which act as sensors and are stacked in between 11 iron plates of 4 metre x 4 metre size. The iron plates are magnetised by passing electricity through copper coils wound around. This is expected to serve the purpose of understanding the engineering issues in constructing the main ICAL, and at the same time provide important inputs on the ICAL's operating parameters and physics measurement capabilities. E-ICAL with 320 RPCs is planned to be setup in Madurai, India. Max throughput expected for E-ICAL is around 200MBps with 10% hit rate and 10k trigger rate.

SYSTEM OVERVIEW

The system consists of several sub-systems: RPC DAQs, Backend Data Acquisition System (BDAQ), Trigger System, Calibration System (CAU), Magnet System, Gas System, and LV/HV System as shown in Fig. 1. Description of each system is beyond the scope of the paper [2].

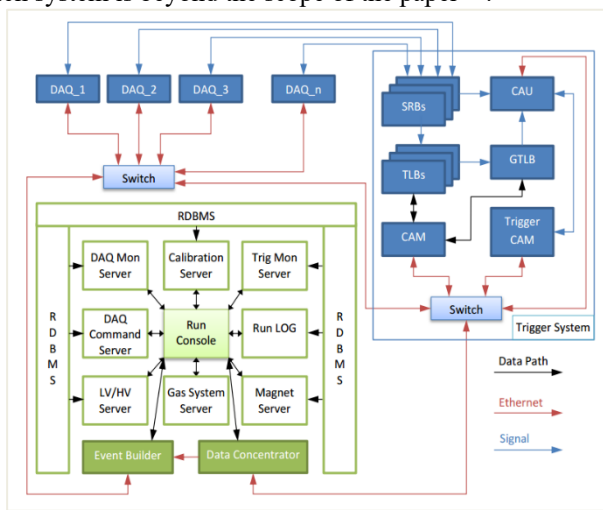


Figure 1: System overview.

Neutrino interacts with the iron plates along its line of travel, triggering events in several RPCs along its path. Orthogonal strip channels (X&Y) on RPCs pick up the charged particles, which are produced from the interaction of neutrino with iron plates. RPC-DAQ modules are connected in hybrid network topology to backend system. Trigger System detects events of interest and notifies RPC-DAQs to transmit event data event data which consists of strips hit and timing information over TCP socket to the designated Data Concentrator (DC) node. Data Concentrator nodes collect event data packets from all the triggered RPC-DAQs and assigns the timestamp and Event Number to the data packet. The updated RPC-DAQ data packets from the data concentrators are pushed to event builder

Backend Data Acquisition System (BDAQ)

The BDAQ system as shown in Fig 2. comprises of several subsystems that are intended to acquire event data and monitor data from the RPC-DAQs. The system also provides event building, event display, data quality monitoring, data archival mechanisms and run manager. BDAQ is a distributed system consisting of several subsystems; Data Concentrator, Event Builder (EB), Run Manager, Data

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Quality Monitoring, Data Visualization, Long term data archival. The scope of the paper is limited to the development of Event Builder module.

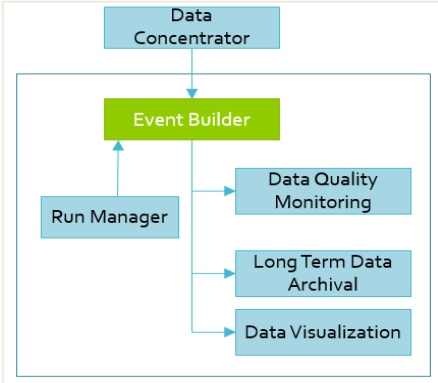


Figure 2: Backend Data Acquisition system.

EVENT BUILDER

Each muon interaction triggers several RPC events; Event builder node is responsible for collating the individual RPC event data packets based on the event number and storing the built-event collection in the required data format. The main functional requirements of Event Builder software include:

- Event data acquisition from Data Concentrator;
- Monitoring of data from RPC DAQs;
- Event Collation from the RPC event data;
- Local data archival in the selected data format;
- Pushing the collated event data to remote consoles;
- Online muon track visualization.

Software Architecture

The software has been developed following layered architecture as illustrated in Fig. 3. Communication layer implements Asynchronous TCP/IP module for receiving variable length RPC data from Data Concentrator, Data push module for forwarding the complete built event to the other remote nodes. The event building layer consists of Data Serializer that parses the data bytes to create valid message object which is binned into a concurrent data structure based on the event number. A high-performance lockless data structure has been used for event collation.

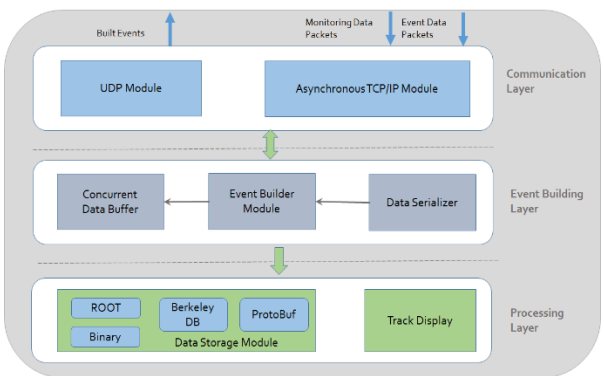


Figure 3: Software Architecture.

The collated event is published to Server push module and processing layer. Processing layer handles the storing the event in the required data format and displaying the track information. Data storage module implements different storage schemes; ROOT, binary, Berkeley DB, ProtoBuf and XML.

Core Modules of EventBuilder

Data acquisition module Data acquisition provides high performance, catering to the system data throughput requirements. The module is optimized to eliminate any possible memory allocation overheads.

Event building Event building algorithm is crucial part of the software. Event building module collates the complete event from out of order RPC data. The module design implements the necessary concurrency control mechanism to handle multi-threaded data acquisition.

Backend data store The collated event is processed by back end store, archiving the data locally. Data storage module write throughput should satisfy required system throughput, otherwise it would create back pressure in acquisition pipeline.

Server push module Server data push module publishes the data to the interested subscriber nodes like data quality monitor consoles and data visualization consoles.

Data Acquisition Module

In general, there are several IO models followed for developing network application, which are broadly divided into thread based and event based models.

Threaded Server Creating a thread per connection with blocking IO calls is the simplest solution for server application. This multi-threaded approach provides concurrent processing of requests. This approach is not scalable due to limited CPU/Memory resources

Synchronous non-blocking IO consists of a single threaded event loop waiting for the readiness of the multiple subscribed socket handles, and triggered socket events will be synchronously dispatched to the corresponding event handler. It is based on synchronous event demultiplexing mechanism. Although this approach can handle more number of sockets than thread-per-connection, but still this is not scalable.

Asynchronous IO is based on asynchronous event demultiplexing. The operations are initiated asynchronously by the application and they run to completion within the I/O subsystem of the OS. Once the asynchronous operation is initiated, the thread that started the operation becomes available. The completed events are inserted into completion event queue and Asynchronous event de-multiplexer removes the completed events and returns to the caller by invoking the corresponding completion handler. Concurrent Asynchronous Event Demultiplexer can improve the performance by making use of thread pool for demultiplex and dispatch completion handlers concurrently [3] [4][5].

The data acquisition module has been developed using multi-threaded *async* IO and is a state full TCP server that keeps track of all the clients connected to it. To reduce memory allocation/deallocation overhead, the server has a

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV013

pool of *SocketAsyncEventArgs*, which reuses the same object for every receive. Buffer Manager handles allocation with dynamic resizing keeping a pool data buffers. The complete message received is propagated to the next module in the pipeline through event trigger mechanism. The module has been implemented using F#. *Async* workflow in F# helped to improve code readability avoiding traditional call back approach. The use of functional features like Active Patterns and Pattern Matching improved expressiveness.

Event Building Module

Event building is a multi-producer single consumer problem, where many RPC message events are being pushed simultaneously. The main responsibilities of the module include;

- Out of order individual RPC events need to be collated to form the complete event.
- Having a provision for accommodating the delayed events if any.
- Collated events propagation to the event processor module should be serialized based on the event number

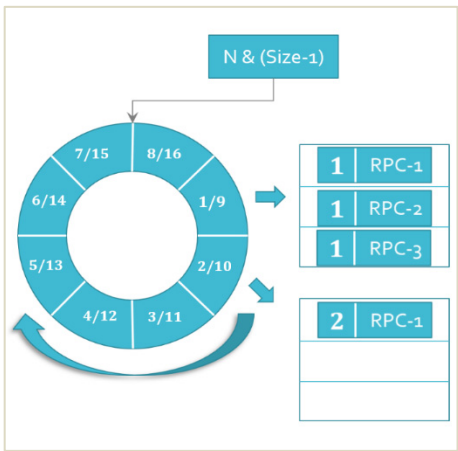


Figure 4: LMAX Disruptor based ring buffer.

Event building module has been developed following LMAX disruptor algorithm [6] as shown in Fig. 4. Disruptor algorithm is based on a bounded ring buffer that allows multiple producer threads without using locking. Each item in ring buffer can hold a list of RPC events, so the ring buffer is a collection of collections. Putting the messages into the buffer is a 2-phase process. First the producer needs to claim a slot in the ring buffer. The sequence/slot number to claim is calculated from the event number, which gives access to a slot in the buffer and the event data is inserted into RPC collection based on RPC ID. All the events with the same event number are collected in the same event slot. The events occupy corresponding entry within a RPC collection based on RPCID. Hence, it does not require locking.

The event slot will be open for event collection until the buffer has no available slot or a timeout occurs. This gives provision for accommodating any delayed events. This event collation process continues for remaining events. A new event claims an occupied slot by publishing the event occupying the slot and updating the cursor variable. Cursor variable tracks the next slot number available for event processing. The event processor handler will be notified which copies the published event data. The algorithm ensures the collated events are propagated to the event processor in order based on the event number. The size of buffer and timeout is calculated based on event trigger rate

Backend Datastore Module

The following are the desirable features of backend data store

- Efficient data access, particularly data write speeds need to match with the max system throughput, as it would create back pressure in the system pipeline
- Cater to large sets of data to suit physics runs which continue for several hours
- Lightweight and embeddable to prevent inter process communication overhead
- Efficient object to event byte stream encoding to save disk space
- Columnar data access/vertical storage techniques that enables retrieving only the selected fields from event record without loading the complete event object
- Forward and backward message format version compatibility as the schema is likely to change over the time, but the archived old format data should still be readable
- Provide extensive querying and data analysis features

Data storage schemes The following data storage schemes were studied and implemented in the software.

- Binary Serialization: Event object is serialized to data byte stream including meta data describing the class. This is the simplest solution [7][8].
- XML: XML Serialization converts object public fields and properties to readable xml stream, which creates a verbose xml file [9].
- BerkeleyDB: This is a light weight embeddable data management library. The memory footprint is just 300KB, but can manage the databases up to 256TB in size. It is a NoSQL database based on key-value pairs. It runs in the address space of the application, hence no communication overhead and all the data access methods are defined through function-call interface [10]
- Google Protocol Buffers(ProtoBuf): It produces faster and smaller byte stream as schema is defined external to the data. It is based on base128 varint encoding. It can support data versioning [11]
- ROOT: CERN developed ROOT framework [12] has been used for event data archiving in mini-ICAL system. It is an Object-Oriented framework for large scale data handling applications. It provides efficient data storage and access system designed to support huge structured data sets. The main reason for choosing ROOT as a backend is its edge in analysis and visualization features.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Server Data Push Module

There are other remote consoles that require the built events for track visualization and data quality monitoring. The server proactively pushes the data to the interested nodes without clients polling for the data. This supports implementation of remote consoles in a technology agnostic manner; browser based, web applications, desktop application or mobile application depending on the usage. This is met with the following design approach as illustrated in Fig. 5.

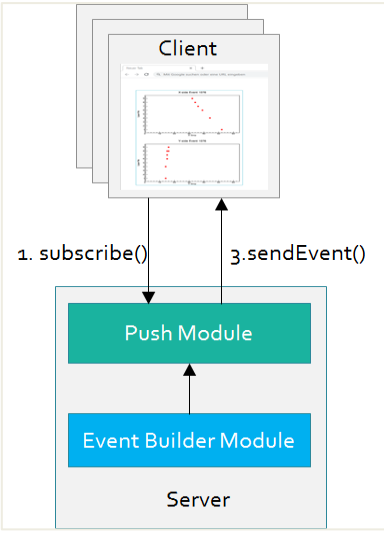


Figure 5: Server data push module.

The remote consoles get subscribed to the back-end server publisher module for various types of events and the server manages the clients into groups based on the subscription type. When the data is ready, the push module will publish it to all the subscribed nodes.

Instead of developing the transport mechanisms for HTTP/TCP from scratch, open source server push technology SignalR [13] is considered. SignalR creates persistent connection between client and server and pushes the server content to clients instantly. It is an abstraction over http and TCP protocols. It supports the following transport techniques for handling real-time communication; http transports websockets [14], SSE, Long polling. Out of which, websockets provide bidirectional persistent communication. if websockets is not available, SignalR falls back to next transport method, based on the capabilities of the server and client

PERFORMANCE

Event builder with all the integrated modules (acquisition, data serialization, event building, event store modules) has been tested for performance with various data storage types. The test setup consists of multiple DC simulator clients sending data to EB server on 1 Gbps Ethernet link. It was observed binary data and BerkeleyDB write throughput was considerably better than ROOT as shown in Fig. 6. With the designed ROOT TTree structure, the write speed observed was around 8MBps due to the write overheads (file meta data). With mini-ICAL, the maximum throughput requirement is around 3MBps. However, due to efficient columnar data access and data visualization, ROOT has been used for data archiving.

Storage Type	Throughput
Binary Data	~90MBps
ROOT	~8MBps
BerkeleyDB	~90MBps

Figure 6: Performance.

TESTING & INSTALLATION

The software was functionally tested with Data concentrator simulator & Front End Electronics (FE) simulator at BARC lab with all the modules integrated to verify the throughput and data validity. The software was installed at IICHEP, Madurai for mini-ICAL. It has been recording data since 2017.

CONCLUSION

The modules have been developed with scalability as design concern. Asynchronous IO based data acquisition module is scalable with the increasing the number of connections. Disruptor algorithm based event buffer provides lock-less data structure for event building. The optimized data structure can be scaled to work for E-ICAL by tuning the buffer parameters.

Non-blocking multi-threaded networking and event building module has been tested up-to 90MBps on 1Giga bit Ethernet network. The required network throughput (200MBps for E-ICAL) can be achieved by upgrading the hardware resources (like 10 Giga bit network and SSD drives). For improving the data write speeds, multi-threaded file writing, multi-level data writing and high performance NoSQL databases like BerkeleyDB will be explored.

ACKNOWLEDGEMENTS

Authors would like to thank the colleagues from TIFR, Mumbai for their suggestions and guidance.

REFERENCES

[1] INO Project Report, vol.1, 2006. http://www.ino.tifr.res.in/ino/openReports/INOReport.pdf

[2] Gobinda Majumdar, Suryanaraya Mondal, "Design, construction and performance of magnetised mini-ICAL detector module", *in Proc. 39th International Conference on High Energy Physics (ICHEP2018)*, Seoul, Korea, Jul. 2018, Paper ICHEP2018_360, doi: 10.22323/1.340.0360

[3] http://www.flounder.com/asynchexplorer.htm

[4] Irfan Pyarali, Tim Harrison, Douglas C. Schmidt, Thomas D. Jordan, "Proactor; An Object Behavioral Pattern for Demultiplexingand Dispatching Handlers for Asynchronous Events", in *Proc. 4th annual Pattern Languages of Programming Conference*, Allerton Park, Illinois, USA, Sep. 1997.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV013

[5] https://docs.microsoft.com/en-us/windows/win32/fileio/i-o-completion-ports

[6] Martin Thompson, Dave Farley, Michael Barker, Patricia Gee, Andrew Stewart, "Disruptor: High performance alternative to bounded queues for exchanging data between concurrent threads", May 2011, http://lmax-exchange.github.io/disruptor/files/Disruptor-1.0.pdf

[7] https://docs.microsoft.com/en-us/dotnet/standard/serialization/

[8] https://docs.microsoft.com/en-us/dotnet/standard/serialization/binary-serialization

[9] https://docs.microsoft.com/en-us/dotnet/standard/serialization/xml-and-soap-serialization

[10] Michael A. Olson, Keith Bostic, Margo Seltzer, "Berkeley DB", Proceedings of the FREENIX Track, 1999 USENIX Annual Technical Conference, Monterey, California, USA, June 6–11, 1999.

[11] https://developers.google.com/protocol-buffers

[12] https://root.cern/

[13] https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-3.1

[14] V. Pimentel, G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket," in IEEE Internet Computing, vol. 16, no. 4, pp. 45-53, July-Aug. 2012, doi:10.1109/MIC.2012.64

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV014

CONTROL SYSTEM OF A PORTABLE PUMPING STATION FOR ULTRA-HIGH VACUUM

M.Trevi†, L.Rumiz, E.Mazzucco, D.Vittor, Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

Particle accelerators operate in Ultra High Vacuum conditions, which have to be restored after a maintenance activity requiring venting the vacuum chamber. A compact, independent and portable pumping station has been developed at Elettra to pump the vacuum chamber and to restore the correct local pressure. The system automatically achieves a good vacuum level and can detect and manage vacuum leaks. It has been designed and manufactured in-house, including the mechanical, electrical and control parts. By means of a touch screen an operator can start all the manual and automatic operations, and monitor the relevant variables and alarms. The system archives the operating data and displays trends, alarms and logged events; these data are downloadable on a removable USB stick. Controlled devices include two turbomolecular pumps, one primary pump, vacuum gauges and one residual gas analyser. The control system has been implemented with a Beckhoff PLC (Programmable Logic Controller) with RS-485 and Profibus interfaces. This paper focuses in particular on the events management and object-oriented approach adopted to achieve a good modularity and scalability of the system.

INTRODUCTION

Sometimes sectors of the accelerator vacuum chamber need maintenance or updates requiring venting, but when they have been carried out it is necessary to create ultra-level vacuum conditions to go back to normal operations.

At Elettra, ultra-level vacuum is usually created locally using a pumping station with on electromechanical logics on board (relays, mechanical timers, etc.). The main disadvantage is that it is not programmable and in case of failure it can stop without any information about the reason (for example a mains interruption due to a thunderstorm). Another disadvantage is that, due to its dimensions, the system is not easily moveable inside the accelerator tunnel. These issues led us to design an automatic and autonomous system managing entire parts of the system: alarm management, operator interface, archiving of variables and events like commands, value changed, etc. An important requirement was the compactness of the entire system that has been achieved by suitable choices of mechanical, electrical and automation components. Moreover another feature that we wanted to reach was the possibility to record log and trend.

PLC, I/O boards and wirings are contained in a 3-unit rack (~180 mm height). The controller based on a Beckhoff CX5120 PLC is compact with very good heat dissipation. The Beckhoff development environment feature an IDE and OOP (Object Oriented Programming) capabilities. An Exor panel has been chosen as HMI.

For the development of the software we have taken inspiration from a template on the Beckhoff site based on OOP [1]. We chose to create OOP syntax in order to be independent on that of the manufacturer. In this way it should be possible to move the software to other controllers. We have assumed that these controllers implement structured text and that a function block can be divided in actions.

The OOP paradigm is mainly based on three principles:

- *Encapsulation*: the internal state of an object can be modified by a public method.
- *Inheritance*: a child class can derive and redefine methods from a super class. The inheritance defines a hierarchy between classes; the mechanism is static and defined at compilation time.
- *Polymorphism*: it is a dynamic mechanism used at run time. If a class has a method *m()* and one subclass (child) redefines *m()*, the polymorphism allows executing an *m()* version according to which kind of subclass object is calling it.

In the PLCs software classes are implemented by Function Blocks (FB), as defined in the IEC 61131-3 standard [2]. The instance of a FB is equivalent to the object of a class. The standard mentions typical keywords of OOP like *methods*, *extends*, *implements*, etc. We have not used these elements in order to comply as much as possible with other kinds of controllers that do not implement these keywords. *Methods* are replaced by dividing the FB code in several independent parts called *actions. Extends* are replaced by writing FB I/O interfaces in order to implement the same concepts.

Another important aspect of the design was alarms and commands management. An alarm can be configured in order to allow the following actions: auto-reset, acknowledge and recording. A command can be recorded by means of the PLC logging system or by the *audit trail*, which is a logging mechanism provided by the HMI. In this log the HMI writes setpoint changes, commands sent to the PLC and it tracks who did what. This feature is widely used in industrial systems [3]; adopting it is particularly easy to save and move log files outside the system for data analysis.

All the system configurations can be easily changed many times during the design and commissioning of the software using a form which we have designed in C# VS2015. The result of this form is written in an XML code to be imported in the file system of the HMI project.

† massimo.trevi@elettra.eu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV014

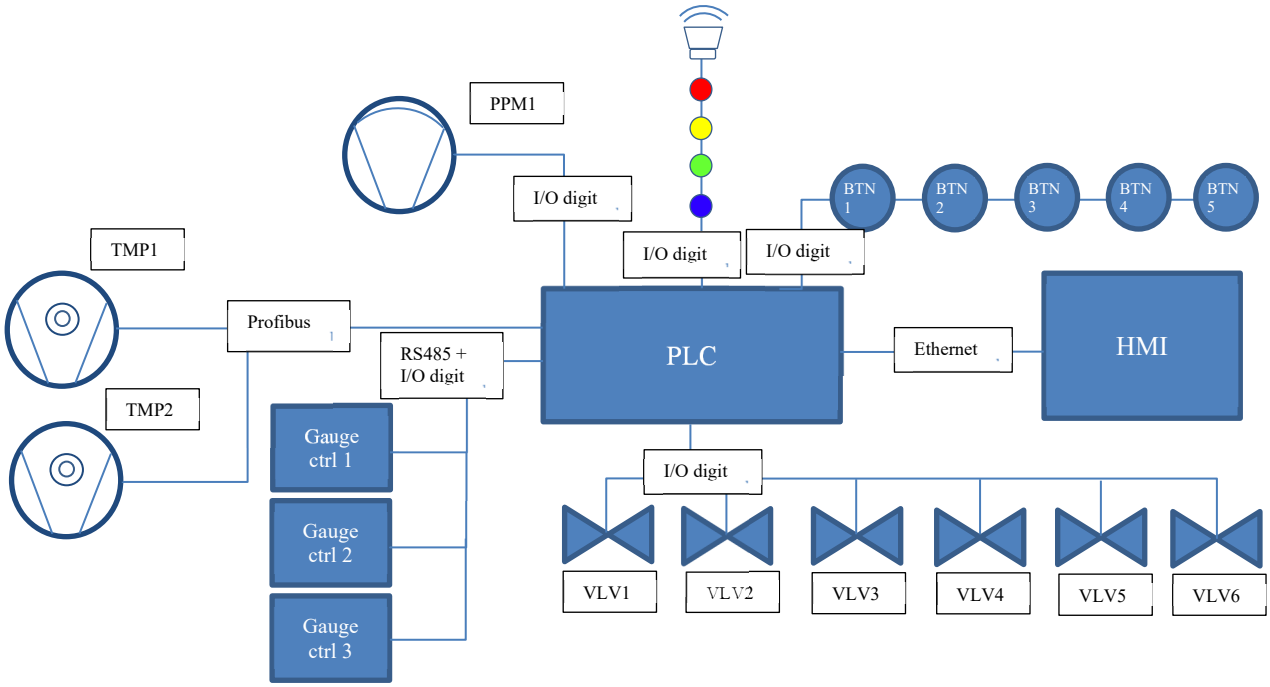


Figure 1: Architecture.

BRIEF DESCRIPTION OF PROCESS

When a vacuum chamber is installed in the accelerator it has to be brought to ultra-high vacuum conditions (~ 10^{-9} mbar). This process is divided into three steps, corresponding to three different vacuum levels and three different kinds of vacuum pumps to be used:

- *Primary* pump (Primary Pump Motor - PPM): starts creating flow vacuum;
- *Turbomolecular* pump (Turbomolecular Pump Motor - TPM): can create high vacuum, in the range 10^{-6}-10^{-8} mbar;
- *Ionic* pump (SIP): allows reaching the target of ~10^{-9} mbar.

The system described in this paper uses the first two kinds of pumps. We have implemented following two features:

- *Leak test*: the test to search for leaks in the vacuum chambers. It is performed by means of a leak detector connected to the vacuum chamber and using helium as a tracer gas; the helium gas has a very small monatomic molecule that can easily penetrate even the smallest holes that the external atmosphere put in contact with the internal vacuum;
- *Residual Gas Analyser* (RGA): instrument that analyses the residual gases inside the vacuum chamber once it has been pumped to a given level. It provides information on the presence of any vacuum leak or contaminants, and on the quality of the vacuum chamber cleaning. Moreover, during some laboratory or industrial processes, it can identify and follow the trend of the partial pressures of the gases introduced into the system (pure gases or mixtures).

The system can be switched in manual and automatic mode. In manual through to the HMI it is mainly possible to command valves and pumps and to set the device configurations. The system will normally be used in automatic mode: this means that when an operator pushes the automatic start button, the PLC automatically sends commands to valves and pumps in order to reach the target pressure.

ARCHITECTURE

Figure 1 shows complete architecture of system. Hardware and software details following.

Hardware

The PLC communicates via Profibus with the turbopump controllers (TMP) and via RS485 with the controllers of the vacuum gauges. The HMI screen is interfaced via Ethernet and the other devices (primary pump, valves, buttons and traffic light) are connected via digital I/Os. In addition, the gauge controllers provide threshold contacts to warn the PLC of any anomaly on the pressure with faster response time.

The traffic light has the following components and meanings:

- *Buzzer*: activated when at least one alarm has to be acknowledged;
- *Red light*: activated when there is at least one alarm;
- *Yellow light*: activated when there is at least one warning;
- *Green light*: if continuous it means that the machine is in automatic mode, if blinking it means that the cycle has started;
- *Blue light* - means that the machine is operating in manual mode.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PLC Software

The main idea has been to divide each task in a number of smaller subtasks. This principle is widely adopted for example for FBs and commands that a device has to execute.

The pumping system is divided into zones which have a logical and process meaning. As shown in Figure 2 the system is divided in 5 zones: *A*, *B*, *C*, *D* and *E*. A zone is defined as the part of the pumping system that can be sectioned by valves. Only the main PLC program can interact with the zone I/O interfaces, which can include other devices interface of FBs like sensors and pumps. In this way this implementation allows to comply with encapsulation.

The main code of this kind of FBs is empty because they are divided in actions. Every action has a task which can:

- be executed by an external command; in this case we call it a *method* or
- expose FB object states to the program or
- interact between internal actions.

The action that implements a command has three features compliant with the method principle:

- call derived FB (class);
- can add code after calling derived class if necessary;
- does not call the derived class in case of rewriting action (override).

These features allow the implementation of *inheritance* and *methods override*.

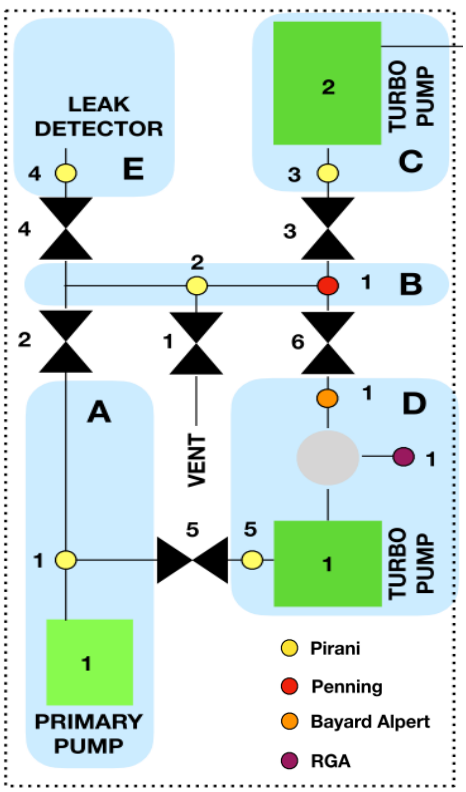


Figure 2: P&Id Process and Instrumentation Diagram.

OBJECT MODELING

The UML-Class Diagram has been used to model the machine devices.

UML (Unified Modelling Language)

UML [4, 5] is a general-purpose, modelling language in the field of software engineering, which is intended to provide a standard way to visualize the design of a system.

A class diagram is a type of static structure diagram that describes the structure of a system by showing:

- classes;
- attributes;
- operations (or methods);
- relationships between objects.

In the Figure 3 we can see that the boxes (FB/class) are connected by different kinds of arrows and lines:

- dashed line with arrow means that there is an association like a dependency between classes; the depend-

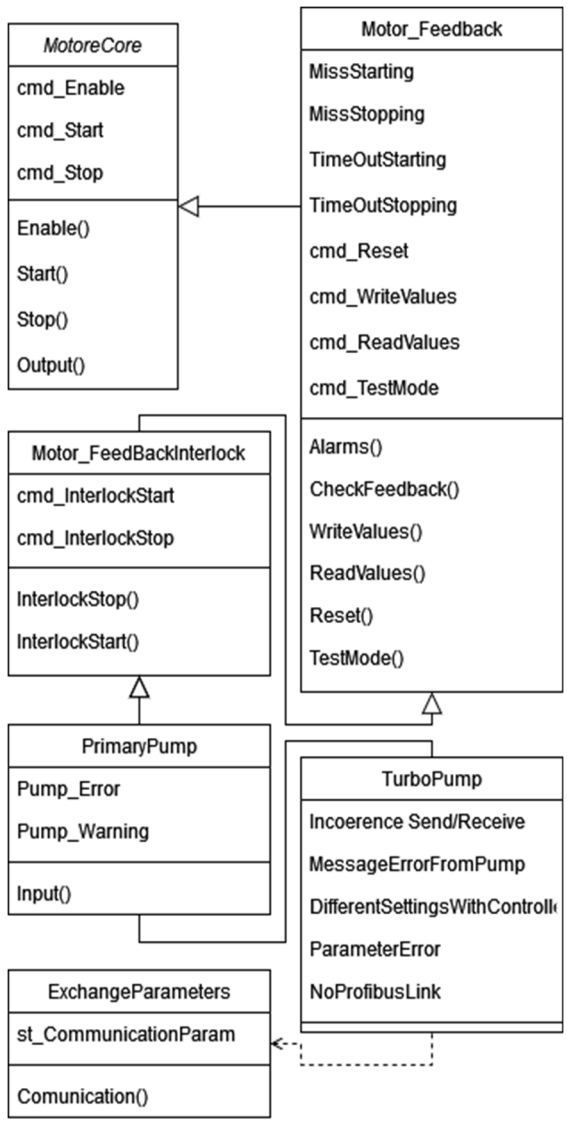


Figure 3: UML - From MotorCore To Pump.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV014

ent class object might use an object of another class in the code of an action/method;
- solid line with arrow means that there is a generalization between FB/class, so there is an inheritance.

The boxes are divided into three parts: on the top we have the name of FB/class, in the middle we have the attributes managed by FB/class, at the bottom we have the actions/methods.

Devices Modelling Example

In order to respect the OOP paradigm, the design of machine device FBs starts from a generic FB (superclass) which is then adapted according to the requirements.

Figure 4 describes the machine model where we can observe the OOP paradigm application in a logical class.

Figure 3 shows the pumps description and the derivation from a generic FB (superclass) of two derived classes: primary and turbomolecular pump FB. They are derived from *Motor_FeedBackInterlock*, which means that primary and turbomolecular pumps have interlock features to start and stop, but also *Motor_FeedBack* and *MotorCore* features.

We would like to underline that a simple push button can be modelled by an OOP paradigm. In our case we have a button with a LED frame and therefore we start from a button that has the electric input (*ButtonCore*) and we derive the *ButtonLight* that has a feedback too.

Figure 4 shows how the machine has been modelled: the machine is divided in zones. A zone can be enabled by a command as well as a device. Every zone has at least one *Pirani* sensor and so *Zone_Pirani* has been derived from *Zone_Core*. From this point there are other three derived FB/classes in order to respect their composition. As explained in the UML chapter different kinds of lines and arrows show different relationships between classes.

Note that, for example, *Zone_Pirani_TP* is derived by *Zone_Pirani* but depends on *Turbopump* class that is modelled in Figure 3.

We can also see this modelling from a P&Id (Process and Instrumentation Diagram) schematic as in Figure 2. In this way we can check matching between a mechanic based scheme and a UML scheme.

FB I/O INTERFACE

Every FB interface that describes something to be displayed on HMI has been designed in order to have the same interface both for a logical or physical type. In addition, it was decided to have a minimum common set of commands and states to be managed in the same way at HMI level.

Having the same interface is useful to organize information to and from a FB. We have divided them in:

Inputs:
- Physical input, connected to PLC boards;
- General input: Boolean, numbers and strings;
- General input: Boolean, numbers and strings used in test mode;

Output:
- Physical output, connected to PLC boards;
- Logical status: represents the status of objects described by FB;

In/Out:
- Alarm;
- Command: corresponds to executing an action/method;
- Data: every kind of number that represents for example thresholds, timers, parameters, configuration, etc.

In/Out variables have been necessary to allow FB/classes to change variables value both inside and outside the code. Note that an Input variable has a part dedicated to TestMode which means that this variable is used when a simulation is running and the object is in TestMode. Simulation is very useful before the commissioning to test automatic cycles, behaviour in case of alarms, etc. Another point to take into consideration in this kind of interface organization is that in the code you can manage devices or logical parts in the same way: they have enable commands, read/write value actions, alarms

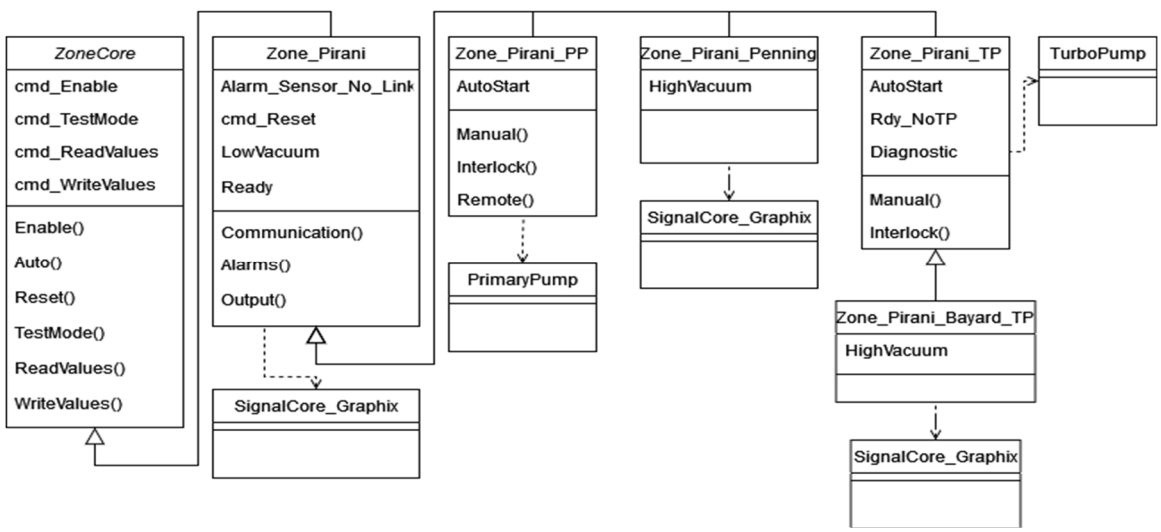


Figure 4: UML Machine Model.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

and warning statuses.

Regarding the second choice, in order to have a minimum common set of states and commands, let's consider for example that we want to reuse the management of an object visualization and therefore, according to the status of an object, display the following states common with different colours:

- *Alarm* in red;
- *Warning* in yellow;
- *Run* in green;
- *Testmode* in blinking blue;
- *Disabled* in light grey;
- *No link* in blinking yellow.-

ALARM AND COMMAND SETTINGS

Another important design choice was how to manage the setting of alarms and commands. In fact, every alarm and command is represented by a 16-bit word to define the configuration. One of the mandatory requirements of the project was the possibility to log and record alarms and actions. Also, anomalous events must be recorded, for example a short black out.

The idea is to have a central logging function in the PLC that is called each time an event happens; examples are a button pushed by an operator or a device that changes state to alarm. An alarm can require an acknowledgement by an operator or be reset automatically. Moreover, an alarm or a command can be logged/recorded. A state associated to each alarm defines if it is active and if an acknowledgement is required. For these reasons the 16-bit word representing an alarm or a command is divided into two parts: 12 bit for the configuration, 4 bit for the state. In this way each alarm and command is represented by a number from 0 to 65535.

While logging is managed by the PLC for example with a circular buffer, the recording action is more complex. To do that we have used the audit trail feature of the HMI, while another feature of the HMI has been employed to acknowledge alarms. For this purpose, a graphical interface using a form written in C# VS2015 has been developed. The form analyses the dictionary created by the HMI. The dictionary is a XML list where tags are described by name and address information. There is a table where it is possible to put the tag name to be found and its setting number. There are some buttons to create tag and alarm files in XML format. According to this number the code configures alarms and tags with the appropriate XML setting based on the specific IDE language. Figure 5 and Figure 6 show an example of setting values contained in a Windows form and the effects on the XML code; the code 52224 is used to configure an alarm to be resettable and acknowledgeable.

```
<requireAck>true</requireAck>
<blinkTxt>false</blinkTxt>
<requireReset>true</requireReset>
<actions>
    <macroAction actionFunction="setBit" parameters="ETOP7M/GVL/g_st_Alarms/st_Zone_C/st_TP/ui_MissStarting;
</actions>
<useractions/>
```

Figure 5: XML extract.

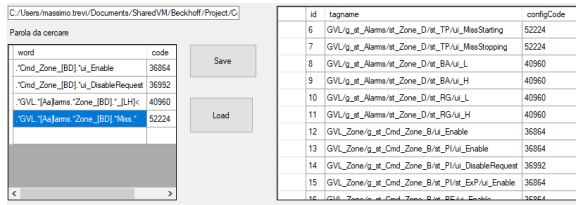


Figure 6: Part of Windows Form.

FUTURE IMPROVEMENTS

There are a number of features that have not been implemented yet because this system is a prototype, but will be taken into account in future implementations.

For example, it would be useful to provide a TCP/IP interface to receive commands and send data via *Tango* for remote panels. Furthermore, there would be the possibility of a *VNC* connection by a remote host with a server on the HMI.

Other features that we want to implement are automatic interlock messages: for example, if we want to open manually a valve that cannot be opened or if the start button is pressed and the cycle does not start, a message should appear to show the reason to the operator. In this way it would be possible to create a troubleshooting system to help the operator to make the right operation to fix the problem and start the system faster.

Moreover, we would like to consider converting the present PLC program in order to use Beckhoff OOP keywords and eventually evaluate improvements in terms of complexity and performance in the two solutions.

Finally, it would be useful to design a superclass for generic serial communications to allow deriving from it the Profibus and RS485 classes.

REFERENCES

[1] Beckoff, https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/2527303819.html&id=

[2] Wikipedia, https://en.wikipedia.org/wiki/IEC_61131-3

[3] Food and Drug Administration - CFR 21 - part 11.

[4] Wikipedia, https://en.wikipedia.org/wiki/Unified_Modeling_Language

[5] Available: https://www.visual-paradigm.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV015

EPICS BASED HIGH-LEVEL CONTROL SYSTEM FOR ESS-ERIC EMITTANCE MEASUREMENT UNIT DEVICE

M. Giacchini, M. Montis, INFN-LNL, Legnaro, Italy
C. Derrez, J.P.S. Martins, R. Tarkeshian, ESS-ERIC, Lund, Sweden

Abstract

The European Spallation Source (ESS) [1] will be a neutron source using proton beam Linac of expected 5MW beam power [2]. The beam commissioning of low energy beam transport (LEBT) started on 2018 and currently expected to reach to the end of Medium Energy Beam Transport (MEBT). Several diagnostics are installed to characterize the proton beam and optimize the beam matching in radio frequency quadrupole (RFQ) section and rest of accelerator. Among all diagnostics, Allison scanner and Slit-Grid type emittance measurement units (EMUs) will aid by characterizing the beam in transverse plane (both horizontal x and vertical y) in LEBT and MEBT, respectively. Here in this paper the Slit-Grid EMU is explained and the software layer developed in EPICS [3] and realized to orchestrate the entire apparatus and control the different sub-systems will be described.

INTRODUCTION

The emittance measurement unit (EMU) aims to measure the transverse emittance by sampling the transverse phase space. At European Spallation Source (ESS) the Alison Scanner and Slit-Grid EMUs will be used to characterize the proton beam transversely in keV and MeV energy ranges, respectively. The Slit-Grid units, designed and delivered by ESS Bilbao [4], are installed in both transverse directions in MEBT, in order to characterize the beam after RFQ, see Figure 1. According to the baseline parameters, the MEBT will operate with peak current of 62.5 mA, energy of 3.63 MeV and RMS size of 2 mm. The Slit-Grid will be used to optimize matching of the beam to the other sections of the accelerator.

Considering the control system aspect, a single EMU device is composed of different sub-systems (acquisition, motion, etc.) which are harmonized and managed by EPICS, the distributed control system framework adopted as standard for the ESS Project. This article reports the upgraded on low-level and high-level control system.

EMITTANCE MEASUREMENT

The transverse emittance is an invariant quantity describing the distribution of the particle beam in transverse phase space (horizontal x and vertical y).

The RMS emittance formula commonly used is the following:

$$\varepsilon_{RMS} = \sqrt{\langle x^2 \rangle \langle x'^2 \rangle - \langle x \cdot x' \rangle^2} \qquad (1)$$

where $\langle x^2 \rangle$ is the variance of the particle's position, $\langle x'^2 \rangle$ is the variance of the angle the particle makes with the direction of travel in the accelerator and $\langle x \cdot x' \rangle$ represents an angle-position correlation of particles in the beam.

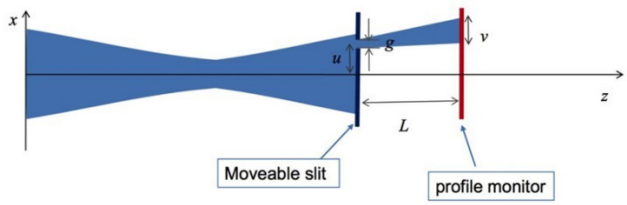


Figure 1: Scheme showing the Emittance measurement using Slit-Grid EMU. Source: Cheymol, R. Miyamoto, *"Preliminary Design of the ESS Slit and Grid System"*

INSTRUMENT DESCRIPTION

A Slit-Grid EMU is composed of a slit and a grid unit, mounted on separate moving actuators to scan the beam entirely. The slit samples small slices from the beam at almost equal position following drift space, the angular distribution of the particles is transformed into a position distribution and sampled using a profile monitor, in our case a secondary emission grid. By scanning the slit across the beam, the whole phase-space is reconstructed, see Figure 2.

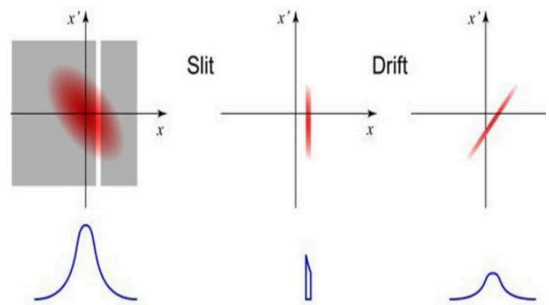


Figure 2: Phase-space sampling using a slit-grid system. Source: B. Cheymol et al., *"Design of a New Emittance Meter for Linac4"*

Particular aspects and design solution adopted are the following [5]:

- For the EMU slit, it was shown that graphite is the chosen material to withstand irradiation. As consequence two graphite plates that form the slit are mounted in the slit head
- The EMU slit is designed in order to scan all the beam aperture. Since the beam envelop is ϕ40 mm,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the blades width has to be >40 mm, for a total height >80 mm. therefore the design for the graphite plates is plates of 40 mm x 54 mmx 3 mm;
- For the slit, the reference values are an aperture of 100 µm and a slit thickness of 200 µm
- The distance between the Slit and the Grid is 400 mm.
- The grid has 24 tungsten wires, 35 µm diameter and the pitch of the grid is 500 µm.
- The resolution of the actuators is at least 25 µm.

The hardware described is shown in Figure 3.

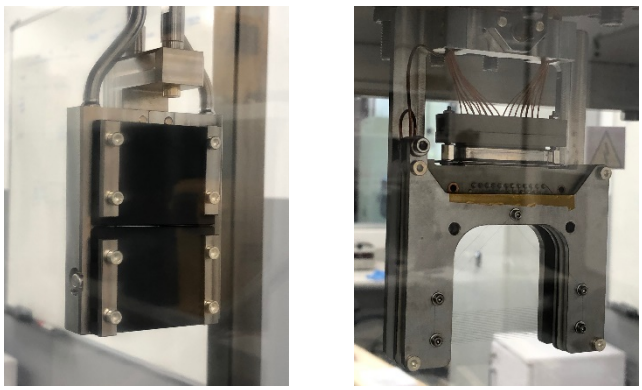


Figure 3: Slit and Grid devices composing the EMU apparatus.

HARDWARE AND SOFTWARE ASSUMPTIONS

The EMU system has been designed in order to satisfy the ESS-ERIC requirements in terms of hardware and software standardization. In particular the system is following these criteria.

Hardware

According to the diagnostic apparatus, it is possible to observe two different functional systems: motion and data acquisition [6].

The EMU motion system is actuated with stepper motors, controlled by a motion controller; slits (horizontal and vertical) and grids (horizontal and vertical) will be stepped across the beam with separate stepper motors (for a total of 4 axes). Two pair of limit switches are installed on each EMU actuator, one for the motion control system, and the other one for the Beam Interlock System.

The EMU data acquisition is devoted to acquire signals from the grid sub-system and provide the information to the high-level software layer designed to elaborate and provide the emittance measurement. The hardware related to the acquisition is composed by an analog front-end for signal conditioning and a digitizer board; in addition to the acquisition chain, a bias voltage supplying line is applied to the grid and an Event Receiver Timing board is required to synchronize the acquisitions with the accelerator global timing.

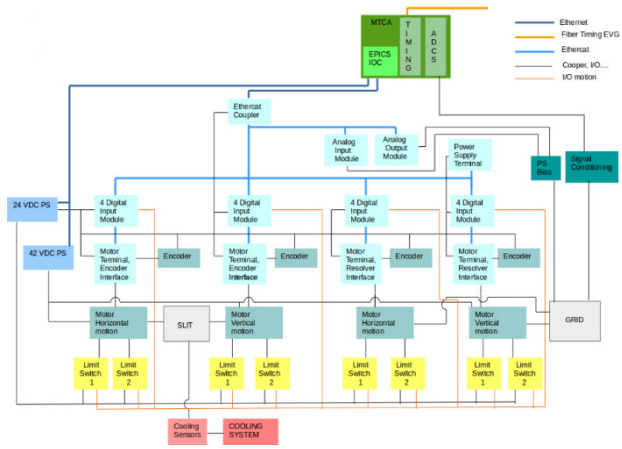


Figure 4: EMU hardware scheme. (Source: I. Mazkiaran, *"EMU Development Document"* [6])

According to the ESS-ERIC guidelines, these two systems are based on Beckhoff® EtherCAT modules [7] for the motion and µTCA® standard for data acquisition [8]. The main schema related to the hardware is shown in Figure 4.

Software

The ESS-ERIC project has adopted the EPICS framework as standard to develop, implement and control the entire facility.

The structure of the control system framework has passed some evolutions in the past years in order to create and provide a solid architecture easy to use and maintain. This aspect is absolutely important in a project where stakeholders' contributions and the final maintenance in charge of ESS-ERIC internal staff are crucial points.

In order to satisfy these requirements, the last solution provided by the project is the ESS EPICS Environment (E3): it is a design concept and a toolkit intended to facilitate development by abstracting away some of the low-level complexities intrinsic to large EPICS implementations (primarily dependency management), and to allow for more manageable quality control of released modules as well as Input/Output Controllers (IOCs). It allows to easily build EPICS modules directly from source and automagically resolves module dependencies, and allows for site specific modifications to EPICS modules without needing to directly modify the source tree [9].

The entire software implemented for the EMU system and described in this contribution is entirely based on the E3 framework.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV015

CONTROL SYSTEM ARCHITECTURE

The EMU device (hardware and software) was originally provided by ESS Bilbao. Under software aspects, the EMU application was not developed using the last E3 environment but with a previous version of the framework. As consequence, several incompatibilities were inherited in the application and a renewal was mandatory; at the same time, additional requirements came up and they had to be developed from scratch. For this reason, a complete software redesign has been done.

The main conditions and requirements considered at this stage were:

- The entire software application must adopt E3 framework. This includes both low-level layer (hardware driver interfaces) and high-level layer (pure software application);
- Slit and Grid axes must be synchronized with data acquisition;
- The algorithm defining the logic in the high-level layer must consider the following set of information as inputs for the emittance measurement:
  - Slit/Grid plane control
  - Slit start position
  - Slit end position
  - Possibility to provide step size or number of steps (per slit position)
  - Grid micro-steps
  - Possibility to operate in *yaw* mode (slit and grid move in sync with a prefixed offset)
  - Number of samples per acquisition
  - Acquisition rate
  - Number of pulses per position
- The algorithm must include the possibility to use pre-calculated slit/grid position instead of providing start and stop positions;
- The entire measurement campaign must be stored in NeXus [10] data format.

Because the software update involves both external contributor (in-kind partner) and internal ESS-ERIC staff, a big constraint in the development stage was related to the necessity to rewrite the entire system application (high-level and low-level software) in parallel by different groups in different places: while the INFN partner was in charge of realizing the high-level control software, ESS-ERIC was in charge of the low-level software migration to the new standard. With this work organization, tasks coordination and teams' communication are crucial to rapidly get progresses.

In order to execute the different upgrades in parallel, a modular approach for the software architecture was adopted: according to the scheme represented in Figure 5, the entire system is composed by 3 main modules:

- Data acquisition (low-level)
- Motion (low-level)
- Experiment supervision and Emittance calculation (high-level)

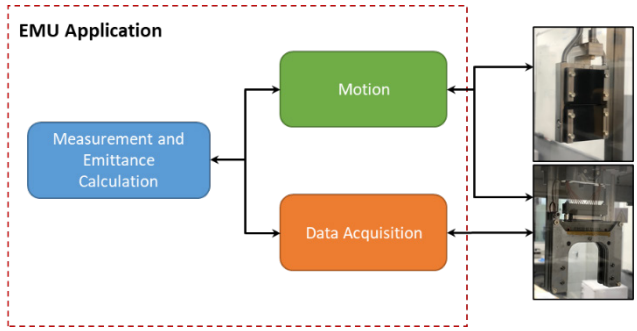


Figure 5: EMU software schema adopted during the application development: it is possible to observe the low-level applications (Data Acquisition and Motion) and the high-level orchestration.

While the driver software interfaces were under upgrade, dedicated simulation mockups were developed in order to provide a set of "black box" applications devoted to simulating the real hardware (Figure 6).

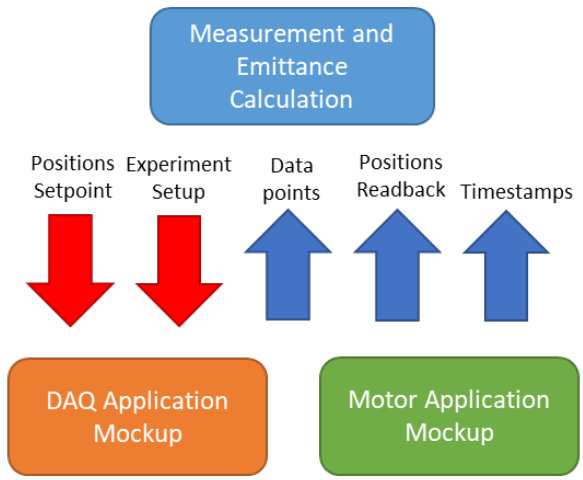


Figure 6: EMU low-level mock-ups used to simulate data acquisition and motion hardware.

LOW-LEVEL SOFTWARE UPGRADE

The data acquisition software layer used in this project is the same that has been used for all the beam instrumentation systems in ESS and it's based in areaDetector drivers. The low-level driver is implemented as an NDArrayPortDriver class, which provides the raw analog data as NDArrays that are further processed by a series of areaDetector plugins. The result of the process chain is the average of the current in each wire of the grid, calculated over a region of interest.

For the motion control, the low-level software is implemented using ECMC [11] framework, the standard platform for the accelerator systems in ESS, built as the software support for the Beckhoff hardware system. ECMC provides the high-level access of the motion system using the EPICS motor record, alongside many other features like the "PLC objects": pieces of code in ExprTK [12] language that are loaded inside the EtherCAT master kernel

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

module and can perform tasks like the automation of the axis homing procedure, protection routines, etc...

HIGH-LEVEL ORCHESTRATION APPLICATION

The high-level orchestration layer is a pure software application and it defines the core part of the emittance measurement. In particular it is in charge of:

- Provide the interface between the system and the user;
- Coordinate data acquisition and motion systems;
- Calculate the beam emittance and all the relative parameters required by the user;
- Collect all the experimental data and store it via NeXus service.

One of the assumptions chosen for the application's design was to create a unique program which does not use any kind of external script or tool. The idea behind this assumption was to provide a standalone portable EPICS application. In order to follow this approach, the high-level orchestration program has been composed by the following parts:

- A set of state machine programs used to coordinate the low-level functional sub-systems;
- A set of EPICS Databases devoted to map the information exchanged between high-level and low-level and between application and final user;
- Additional custom C libraries used to implement particular functions required by the application and not available in the standard one.

The core of the main application is the set of state machines: using the common Sequencer EPICS module, different SNL programs were defined [13]. A main state machine is dedicated to control the system, taking in charge of capturing the measurement scan parameters, execute preliminary checks, coordinate and synchronize motion and data acquisition, elaborate the data coming from the field, execute emittance calculation and send the information of interest (measurement raw data and metadata) to the NeXus service. Secondary state machines have been created in order to operate minor functionalities, such as parameters calculation for measurement setup and optimization (conditions checks).

Because of the estimated size of the data of the measure campaign (number of acquisitions, samples acquired per wire, etc.), one critical point the SNL program had to manage was the allocated memory at application start-up, coming up with the definition of multi-dimensional arrays used to manipulate sets of signals coming from the field. In order to overcome this possible critical point, 2 different elements had been used in the SNL program:

- The signal coming from the DAQ system were preliminary manipulated via EPICS AreaDetector [14] and a set of its plugins: Processing plugin, ROI plugin and Stats plugin (Figure 7);
- Custom C functions devoted to define pointers, data structures and all the calculations required by the measurement campaign. These functions were grouped and managed as local library and inherited in the SNL program through the *Escape to C Code* mechanism.

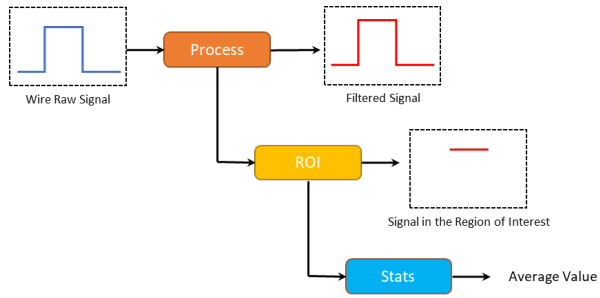


Figure 7: Wire signal processing with EPICS AreaDetector plugins.

In order to store the sensible information (raw data and experiment metadata) in the NeXus service, the ENeXAr tool [15] was used. This application a Python service developed internally at ESS-ERIC that can be used to archive PVs into NeXus files through simple CLI commands; this solution was extremely useful to easily integrate the service inside the orchestration state machine.

As previously mentioned, NeXus service is in charge of storing experimental raw data measurements and metadata using HDF5 file format [16] largely used in research environment. In addition to this storage system, the canonical EPICS Archiver service is used to store all the control variables of interest.

TEST AND FURTHER DEVELOPMENT

Preliminary tests demonstrated that the application requires a minimum 16GB of RAM to be stable. This high request of memory is caused by the number of data structures initialized by the state machine during application start-up.

The first bunch of tests were dedicated to verify the correct management of the functional sub-system (data acquisition and motion) using the simulated environments. For this purpose, the Graphical User Interfaces shown in Figures 8-10 were developed.

Once these tests will satisfy all the projects requirements in terms of functionalities, a second bunch of tests will be performed connecting the EMU application with the real hardware and with the new low-level drivers. In parallel to this task, the main application will be revised with the intention of optimizing memory consumption.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV015

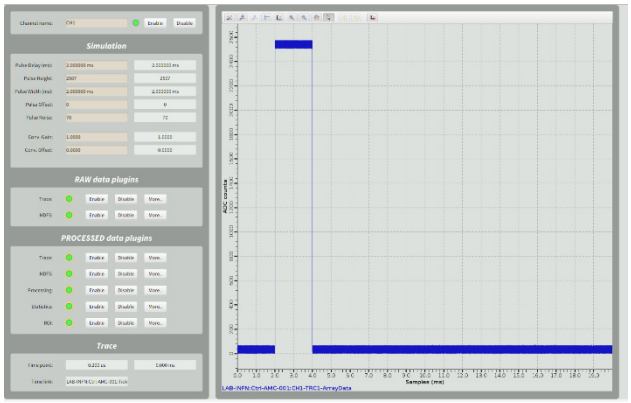


Figure 8: GUI developed for data acquisition system.

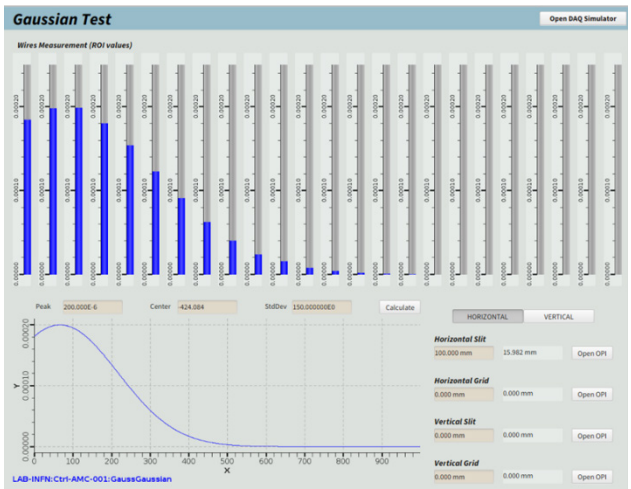


Figure 9: GUI developed for motion system.

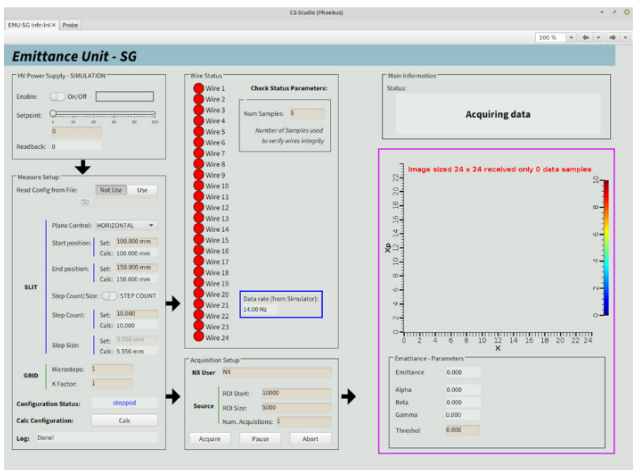


Figure 10: GUI developed for the EMU application.

CONCLUSION

The Emittance Meter Unit is an important equipment for calculating the transverse emittance related to the beam in the MEBT section of the linac. Due to the requirement and indications coming from ESS-ERIC Guidelines in terms of hardware and software standards, the application developed has been a great challenge in the matter of technical solutions adopted during the software upgrade: time optimization had a key role in the entire work.

The application has been tested with the simulated environment and the preliminary results are promising. Further tests with the real hardware are scheduled in the next period. In the meanwhile, code optimization in terms of memory usage will be done.

In addition, the long-term goal the team would achieve is using the results of this work is to establish a scanning instruments EPICS framework for the entire project, reusing it on the EMU systems in the LEBT and on the Wire Scanners instrumentation.

ACKNOWLEDGMENTS

This work can't be possible without the contribution of the colleagues of ESS Bilbao, the Beam Diagnostic Team and the ICS at ESS.

REFERENCES

[1] ESS – European Spallation Source, https://europeanspallationsource.se/

[2] R. Garoby *et al.*, "The European Spallation Source Design", *Phys. Scr.*, Vol. 93, 014001, 2017 https://doi.org/10.1088/1402-4896/aa9bff

[3] EPICS, https://epics-controls.org

[4] I. Mazkiaran, I. Bustinduy, C. de la Cruz, G. Harper, A. Rodríguez Páramo, and J. P. S. Martins, "ESS MEBT Control System Integration", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-MOPHA091

[5] A.R. Páramo et al., "Conceptual Design of the ESS MEBT EMU", July 2018, unpublished

[6] I. Mazkiaran, "EMU Development Document", May 2017, unpublished (Ref . MEBT-BI-EMU82-04)

[7] Beckhoff® EtherCAT System, https://www.beckhoff.com/en-en/products/i-o/ethercat/

[8] µTCA® open standard, https://www.picmg.org/openstandards/microtca/

[9] EPICS E3, http://e3.pages.esss.lu.se/

[10] NeXus project, https://www.nexusformat.org/

[11] J. Etxeberria, J. H. Lee, and A. Sandström, "EtherCAT Open Source Solution at ESS", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-WEPHA046

[12] C++ Mathematical Expression Toolkit Library (ExprTk), https://github.com/ArashPartow/exprtk, https://github.com/icshwi/e3-exprtk

[13] EPICS SNL, https://www-csr.bessy.de/control/SoftDist/sequencer/

[14] EPICS AreaDetector, https://cars.uchicago.edu/software/epics/areaDetector.html

[15] ENeXAr service (internal ESS documentation), https://confluence.esss.lu.se/display/SW/ENeXAr+-+The+EPICS+NeXus+Archiver

[16] HDF5 library and file format, https://www.hdfgroup.org/solutions/hdf5/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV016

DESIGN AND DEVELOPMENT OF THE NEW DIAGNOSTICS CONTROL SYSTEM FOR THE SPES PROJECT AT INFN-LNL

G. Savarese*, G. Arena, D. Bortolato, F. Gelain, D. Marcato, V. Martinelli, E. Munaron, M. Roetta
INFN/LNL, Legnaro, PD, Italy

Abstract

The need to get finer data to describe the beam is a relevant topic for all laboratories. For the SPES project at Laboratori Nazionali di Legnaro (LNL) a new diagnostic control system with more performing hardware, with respect to the one used in legacy accelerators based on Versabus Module Eurocard (VME) ADCs, has been developed. The new system uses a custom hardware to acquire signals in real time. These data and ancillary operations are managed by a control system based on the Experimental Physics and Industrial Control System (EPICS) standard and shown to users on a Control System Studio (CSS) graphical user interface. The new system improves the basic functionalities, current readback over Beam Profilers (BP) and Faraday Cups (FC) and handlings control, with new features such as: multiple hardware gain levels selection, broken wires correction through polynomial interpolation and roto-translations taking into account alignment parameters. Another important feature, integrated with the usage of a python Finite State Machine (FSM), is the capability to control an emittance meter to quickly acquire data and calculate beam longitudinal phase space through the scubeex method.

INTRODUCTION

The SPES project, acronym for Selective Production of Exotic Species, is the new leading project of the Legnaro National Laboratories (LNL) an Italian international center for the nuclear and astronuclear physics research. The objective of the SPES project is to study exotic beams made up of ions much more neutron-rich than those we can find in nature. It will exploit the legacy accelerator ALPI (Linear Accelerator for Ions) and will be employed as another ions source in parallel to the existing ones. [1] Since the beam will be post-accelerated in the ALPI accelerator, the beam line is made up of instruments aiming to optimize the beam transport, to reduce the energy spread, to clean the beam from undesired isobaric masses, to boost the beam charge and to pulse the beam. In a transport beam line the fundamental elements to monitor the beam shape, position and intensity are the diagnostics. They are essential to understand how to modify system parameters to obtain the desired beam [2].

The SPES project was the trampoline for the LNL to upgrade the hardware used to acquire and elaborate data and to adopt the Experimental Physics and Industrial Control System (EPICS) standard [3], that is a set of software tools developed to operate particle accelerators allowing information access and exchange between different subsystems. In fact a general upgrade of the legacy diagnostic control system was needed since the computer technologies improvement and the increasingly challenging accelerator data acquisition requested the design of a new modular and maintainable software and a hardware offering better performances.

This paper shows the most significant details and advantages of the new hardware configuration and the EPICS Input Output Controller (IOC) developed to communicate with the hardware components and the other subsystems already developed at the LNL.

THE LEGACY CONTROL SYSTEM

The legacy diagnostic control system is currently mounted in the ALPI and PIAVE (Positive Ions Accelerator for VEry low velocity ions) facilities and its target is to control and display details related to Faraday Cups (FC) and Beam Profilers (BP). Our FCs are copper cups used to collect and measure all the beam current; a metal shield, charged with high negative voltage, avoids that secondary electrons exit the cup. Our BPs are made up of two perpendicular grids (vertical and horizontal) with 40 parallel tungsten wires. Other diagnostic instruments are the collimators (or slits), used to narrow the beam, and the Emittance Meters (EM), which measures the longitudinal phase space and the energy spread.

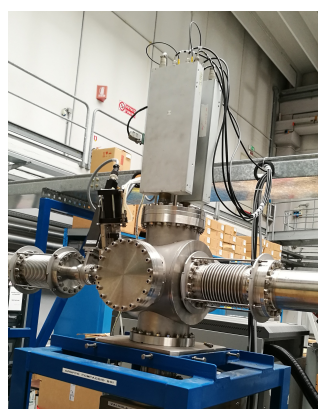


Figure 1: Diagnostic box with the legacy pre-amplifier grids.

Figure 2: VME Rack.

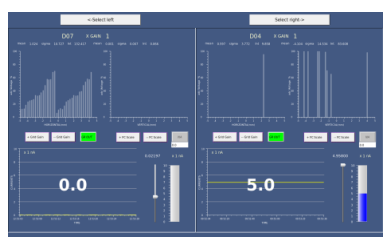


Figure 3: Legacy diagnostic GUI.

* giovanni.savarese@lnl.infn.it

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV016

The electronics, for grids signals, is made up of an analog board, as shown in Fig. 1, mounted close to the detector to reduce the signal path. It has a current to voltage converter for each wire of the grid and a multiplexer system generates an output buffer to send to a Versabus Module Eurocard (VME) ADC board. The current amplification, performed by the converters, is controlled by two transistor-transistor logic (TTL) signals connected to two analog switches; the saturation levels are respectively 10 nA and 100 nA and the lower detectable current is 1 nA [4].

The VME ADC board, Fig. 2, is located outside the bunker to ease maintenance and reduce the exposure of electronic instrumentation to radiations. This board provides 12 bits resolution and supports a conversion rate of 100 KHz. Current signals coming from the FC are directly amplified and digitized. BP signals come as two buffers of 40 voltage signals and the board scans them thanks to a clock signal, composed by 40 TTL period pulses of 200 µs each one, kept 1.6 ms apart. The clock signal and the selection gain signals pass through a digital buffer, suited to drive 50 Ω loads in a daisy-chaining arrangement through subsequent boxes.

At the beginning, the software layer was based on C programs developed on VxWorks Operating System (OS). Then it was replaced by a custom EPICS based control system, hosted by the VxWorks OS with a device driver realizing the interface between the field and the EPICS server host: the IOC receives the BPs and FCs digitized values and apply a software conversion to retrieve the current values [5].

The steppers motor controllers are VME cards built at the LNL, together with the associated power drivers: every single controller is able to manage up to 8 motors. The controller is VME based and is hosted, with the associated EPICS IOC, by a VxWorks OS.

A Control System Studio (CSS) Graphical User Interface (GUI), Fig. 3, has been developed to read the EPICS Process Variables (PVs). The main panel shows general information about diagnostic boxes, such as BPs and FCs status and commands. When selecting a specific diagnostic, users are redirected to a page showing the beam vertical and horizontal profile and the FC current. From the same page, it is possible to graphically adapt the signal full-scale, but not the hardware gain value, and to insert or extract the BP and the FC when presents.

CONTROL SYSTEM UPGRADE

The new control system will be EPICS based and will run on a Linux virtual machine instead of on an embedded OS. It will exploit a more performing hardware and an improved software with new features. Since the control system upgrade will take some time, and in an initial period there will be a mixed regime with both the old and the new control system, the requirements are: both hardware and software must be backward compatible with the exiting ones and modular; to improve the acquisition precision and the position control; possibility to hide broken wires, which usually falsify the measurement, and reorder switched wires; create an improved GUI similar to the legacy one; integration of different type of handling systems.

Hardware

Knowing these requirements, we developed a custom multi input and output purposes controller and a pre-amplifier box [6].

Figure 4: Custom pre-amplifier box developed at LNL.

Figure 5: Custom controller developed at LNL.

In the new configuration, two 40 wires cables start from the same BP and reach the same pre-amplifier box, Fig. 4. They bring information about the vertical and horizontal profile each. The new pre-amplifier box is made up of two board, each with 40 Operational Amplifier (OA). Each OA converts the received current value into a voltage value in the range [-10, 10) V performing a signal amplification based on the selected gain. The selected gain is the same for all the OA in the box. There are 4 gain levels allowing to read current values from 1 pA up to 10 µA. The converted signals, using the clock signal received from the controller, are then multiplexed over a single wire and forwarded to the controller. The new pre-amplifier box can be controlled by both the legacy VME board and the new controller.

To replace the legacy VME crates, we chose to use the controller in Fig. 5 which is a smarter and more cost-effective solution for those applications, not involved in high-reliability processes, where the control tasks are reduced to simple analog and digital I/O operations. The core element is the Field Programmable Gate Array (FPGA) allowing to perform real time tasks, fast peripherals control, data acquisition, digital data pre-processing and data buffering. The configurable digital and analog I/O channels and the external communication through gigabit Ethernet and Wi-Fi make the controller

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV016

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

a complete and improved data acquisition system for real time applications with respect to the legacy electronics. The I/O channels flexibility and the fact that the FPGA can be programmed to generate clock and gains signals, equal to the ones generated by the VME crates, makes the controller backward compatible allowing the communication with both the legacy analog boards and the new pre-amplifier boxes.

Thanks to its modularity, a single controller can manage up to 4 pre-amplifier boxes (equivalent to 4 BPs) and 4 FCs. BPs values arrive to the controller as two buffers of 40 voltage signals and an ADC digitizes them into two arrays of 16 bits values. The ideal conversion is expressed in Eq. (1). These arrays are stored in specific registers of the FPGA. On the other side, the controller uses a CAENels board to read FCs current values. The readable current range spans in the range [-FS,FS), where FS stands for the hardware full-scale imposed by the selected gain which can be 1 mA or 1 µA. The digitized value is saved as a 20 bits value in the FPGA register and Eq. (2) describes the ideal conversion.

$$X_{BP,i} = \frac{V_{BP,i} \cdot 2^{16}}{2 \cdot FS_{BP}} = V_{BP,i} \cdot G_{BP} = I_{BP,i} \cdot G_{PA,i} \cdot G_{BP} \quad (1)$$

$$X_{FC} = \frac{I_{FC} \cdot 2^{20}}{2 \cdot FS_{FC}} = I_{FC} \cdot G_{FC} \quad (2)$$

EPICS IOC

For the software development we chose to use the EPICS framework and we implemented a new IOC from scratch. The IOC reads the data saved in the controller's FPGA, elaborates those data and communicates with external IOCs controlling different types of handling systems. To provide a common interface, during the transaction phase where both the control systems will coexist, a software abstraction layer inside the new IOC communicates with the PVs of the legacy diagnostic IOC and integrates them in the new control system.

Current values retrieval To communicate with the controller, the IOC makes use of IPBUS. IPBUS is an IP-based communication protocol and the transport protocol can be UDP (User Datagram Protocol) or TCP (Transmission Control Protocol) based. At LNL we developed an EPICS device support to implement the IPBUS protocol based on UDP. Through this module we can associate a generic PV to specific FPGA memory register. This way PVs can access acquired data or can change the acquisition mode changing the related control bits. Thanks to these control bits, users can select the gain to apply during the acquisition.

Once the IOC has loaded the raw values, knowing the hardware specifications and given a set of calibration values to compensate the ADC and OA errors, the logic retrieves the correct current value. For BPs, the controller acquires array of values, so the conversion and the calibration is specific for each value, moreover the error contribution comes from both the ADC and the OA. Another feature available when dealing with arrays is the possibility to define a mask to redirect each value to the correct index of the output array.

$$I_{BP,i} = \left(\frac{X_{BP,i} \cdot S_{BP}}{G_{BP}} + O_{BP}\right) \cdot \frac{S_{PA,i}}{G_{PA,i}} + O_{PA,i}[A] \quad (3)$$

$$I_{FC} = \frac{X_{FC}}{G_{FC}} \cdot S_{FC} + O_{FC}[A] \quad (4)$$

On the other side, the abstraction layer, which does not communicate with the new hardware, gets the current values stored in the legacy PVs through the Channel Access (CA). The abstraction layer is used only for those diagnostics still controlled by the legacy control system.

Broken wires In BPs, a frequent problem is the presence of broken wires. Usually a broken wire is a disconnected wire, not measuring current, or a deformed wire touching one of the adjacent wires creating a short-circuit; in the latter case we have two adjacent broken wires showing saturated negative current and saturated positive current each. Given the array of the current values, user can select the index of the supposed broken wire and hide it. When there are broken wires a PV executes a C subroutine to restore the correct value.

Currently users can choose between two methods: the first sets broken wires current values equal to the mean value between the previous and the next valid wire; the second interpolates the array with a polynomial function of grade 7. With this grade, we saw a good approximation of the real values but other tests are still undergoing.

Handling control At the LNL we have different actuators controlled by specific IOCs: stepper motors, pneumatic motors and pneumatic pistons. Given the actuator type, the diagnostic IOC instantiates a set of standard PVs connected to the PVs of the IOC controlling that specific actuator. This way, users can move the correct instrument knowing nothing about the actuator details.

In addiction to the interface with multiple IOCs, the handling logic converts the actuator position from its reference system (r.s.) to the beam line (or absolute) r.s. and vice-versa. This is achievable using a set of roto-translation matrices. More in details we assumed the detector instrument is mounted at the end of the actuator with a certain offset, along the X and Y axis, and a rotation angle along the XY plane. Similarly, the actuator is mounted over the flange with its set of offsets and angles. In the end, the flange is mounted with a specific distance from the center of the beam line and, depending on the box, it has a rotation angle. These three roto-translation matrices have the same shape described in Eq. (5) which represents a standard roto-translation along the z-axis. With these considerations, the correct parameters and the actuator position in its r.s., the logic returns the instrument accurate position represented in Eq. (6).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV016

$$T = \begin{bmatrix} cos(\theta) & -sin(\theta) & O_x \\ sin(\theta) & cos(\theta) & O_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} P_{xb} \\ P_{yb} \\ 1 \end{bmatrix} = T_{fb} \cdot T_{mf} \cdot T_{im} \cdot \begin{bmatrix} P_{xi} \\ P_{yi} \\ 1 \end{bmatrix} \quad (6)$$

On the other side, the abstraction layer, which does not know the alignment offsets, links the *insert* and *extract* commands to the legacy PVs not applying roto-translations. In those cases, the instruments position is nominal.

Collimators control Collimators (or slits) are a couple of metallic plates used to narrow the beam and the main parameters to control are the slot's center and aperture. In contrast to BPs and FCs, that can be only completely inserted or extracted they can reach different positions. There can be multiple types of collimators: two actuators, each with its plate, mounted over the same flange; two actuators, each with its plate, mounted over opposite flanges; a unique actuator, with both plates, controlling only the slot aperture size; a unique actuator, with both plates, controlling only the slot center position.

Considering that each slit has its own roto-translation logic, the control logic has an abstraction layer that, based on the collimators configuration, translates the set and read commands to show a common interface to the end user.

Graphical User Interface

A new CSS GUI, emulating the existing one, with additional features has been developed. The main page shows a list of the available transport elements such as triplets, dipoles and more. Chosen a specific item, users are redirected to the correct page. Since it is useful to monitor the measured current while changing a transport element parameters, diagnostics and transport element pages occupy only half of the horizontal screen size. Figure 6 shows the most important pages: the page showing the BP and the FC details and the page with the collimators position control. Other available pages are: the handling alignment parameters page, the broken wires page and the beam dynamics parameters page (with the parameters used to find the best beam trajectory).

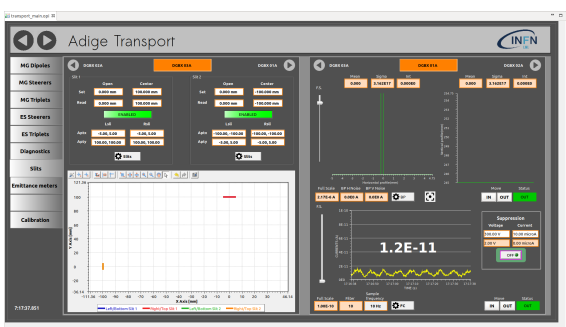


Figure 6: CSS based diagnostic GUI.

The BP section can show the vertical and the horizontal beam profile using two independent graphs or an intensity map; a slider allows to change the graphs full-scale and consequently the hardware gain. Each graph shows the correct position of each wire and the mean point of the current waveform. In addiction the page shows the standard deviation and the total current measured by each profile. The FC section shows a graph with the current trend and a panel to set the suppression level. In both sections the page shows the actuator status and allows users to insert or extract the instrument. Another improvement is the presence of a set of parameters to graphically remove data noise and a set of buttons to open the actuator specific page.

The collimators page shows every slit system mounted in a specific diagnostic box. Two control modes exist: read and set the slot aperture and center; read and set the left and right slit position. Only collimators systems with two actuators have both the control modes, the others can control only the center or the aperture. At the page bottom there is an XY graphs showing the current position of each slit.

EMITTANCE METER

The emittance meter is based on two identical slit-grid type instruments that scan the beam in two orthogonal planes [7]. This method offers very clear results since the beam divergence is scanned directly and simplifies the calculations involved during the data analysis. The slit–grid instrument, consists of a grid (the detector), orthogonal to the beam, formed by 77 parallel wires, uniformly spaced by 1 mm and a plate (the collimator), parallel to the grid and placed 300 mm before the grid, with a slot of 0.3 mm aligned with the grid center. This solution allows for angular measurement in the range ±60.0 mrad. A chassis holds the collimator and the detector and moves them together.

The electronics used to control the emittance meter acquisition and motion is the same used for the diagnostics. The main difference is that each emittance meter grid signal requires two pre-amplier box channels and two controller channels. To reconstruct the correct signal the IOC combines together the signals belonging to those channels and then applies the conversion logic previously described. The motion control follows the same rules defined for the previous instruments.

Acquisition Procedure and Data Analysis

When the device is placed in the beam path, the slit selects the beam *slice* at that position and the grid reveals its divergence distribution. This way, the emittance measurement is performed by gradually inserting the slit across the beam path and measuring the divergence distribution on equally spaced rows, covering the whole beam area.

To perform this procedure the control system uses a python based Finite State Machine (FSM). It is a python script running in background using the *pysmlib* library [8]. This library helps creating event based FSMs, each running on a different thread, for EPICS control systems. The connection through

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV016

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the CA to the desidered PVs is provided by the well known library *pyepics*. The FSM checks its current state every time an input or output PV changes and, if the conditions allow it, the code of the specific state is executed. The library also provides timers for delayed operations which are largely used in this project to check if a motion has been completed.

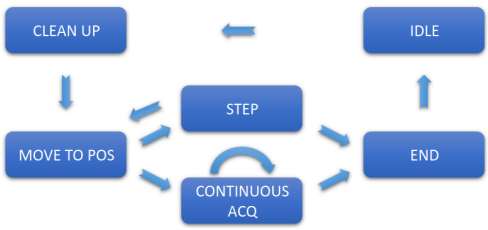


Figure 7: Python FSM to control the acquisition procedure and the ellipse calculation.

In the emittance meter FSM, as described in Fig. 7, the FSM remains in the idle state till the user starts the procedure. The procedure prevents the vertical and horizontal contemporaneous scanning that would lead to mechanical damages or inaccurate acquisitions. When the procedure starts, the FSM cleans up remaining values, moves the grid to the initial position and then starts the acquisition. There are two acquisition modes:

- **Step-by-step**: The FSM reads the current acquired by the grid and store the values in an array. It waits 0.5 s then goes to the next position, based on the number of steps and the ending position. It repeats these steps till the motor reaches the ending point.
- **Continuous**: The FSM moves the motor at constant speed till it reaches the ending position and constantly stores the values acquired by the grid into an array.

The resulting matrix contains, for each position, the array with the current values, acquired by the grid, included in the angular range defined at the beginning of the scan; those current values should return the emittance ellipse. At the end, the motor returns to the homing position (grid completely extracted) and the GUI, Fig. 8, shows the acquired matrix. From that GUI users can choose the acquisition mode (continuous or step by step), set the motion speed, the number of steps, the angles and positions range to consider when acquiring data and start or stop the acquisition procedure.

In some cases the ellipse estimation can be affected by errors since the presence of noise. For this reason the python script uses the scubeex-ghostbuster algorithm [9]. It is a recursive algorithm that estimates the ellipse (the beam current values at each position) and a threshold value (all values below the threshold will be considered noise and removed from the ellipse calculation) till it reaches an asymptote. At the end the *ghostbuster* routine removes the areas affected by the scattering of ions improving the emittance calculation. From the GUI, it is possible to change the threshold value and other parameters to manually find the best threshold value and the best emittance.

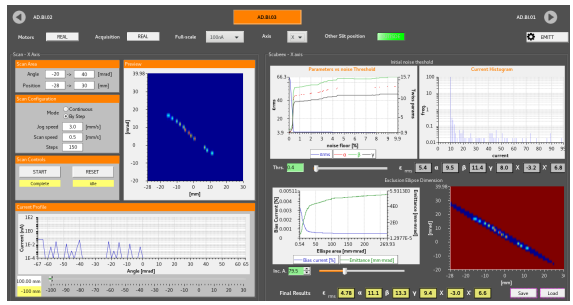


Figure 8: CSS based Emittance Meter GUI.

CONCLUSION

Our first test foresaw the installation of the new hardware and software to control to control one EM and two standard diagnostic boxes with one BP, one FC and one collimator systems each. This bench test was essential to test and improve the control logic. In fact these tests confirmed the improved quality of the hardware reading up to pA values, we successfully tested the EM FSM and operators gladly appreciate the new features making them the starting point for new high level functionalities. Subsequently we successfully started a different instance of the same IOC to test the translation layer for all the diagnostics currently mounted in ALPI. This was the first step for the general upgrade of the ALPI diagnostic control system. Confidants of the excellent results obtained from these tests, the next steps are the substitution of the legacy hardware in ALPI and the installation of the new SPES diagnostics.

REFERENCES

[1] LNL site, accessed on Sep. 2021, http://www.lnl.infn.it/index.php/en/

[2] G. Savarese *et al.*, "Design and development of the diagnostic control system for the SPES project", Apr. 2019.

[3] EPICS site accessed on Sep. 2021, https://epics-controls.org/

[4] M. Bellato, A. Dainelli, and M. Poggi, "The beam diagnostic system of the ALPI post-accelerator", in *Proc. EPAC'92*, Berlin, Germany, Mar. 1992, pg. 1085-1087.

[5] M. Giacchini *et al.*, "Upgrade of the beam diagnostic system of ALPI-PIAVE accelerator's complex at LNL", in *Proc. PCaPAC'14*, Karlsruhe, Germany, Oct. 2014, paper WPO018, pp. 72–74.

[6] D. Pedretti, "Design and Development of a Multi-Purpose Input Output Controller Board for the SPES Control System", Ph.D. thesis, Università Degli Studi di Padova, 2018.

[7] J. Montano *et al.*, "Off-line emittance measurements of the SPES ion source at LNL", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 648, pp. 238-245, Aug. 2011. doi:10.1016/j.nima.2011.05.038

[8] Pysmlib site, accessed on Sep. 2021, https://darcato.github.io/pysmlib/docs/html/index.html

[9] L. Bellan *et al.*, "SCUBEEx-GHOSTBUSTER algorithm for ghost infested emittance measurements", LNL-AR, Italy, 2015.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV019

CONTROL SYSTEM FOR 30 keV ELECTRON GUN TEST FACILITY

D. A. Nawaz†, M. Ajmal, A. Majid, N. U. Saqib*, F. Sher
LINAC Project, PINSTECH, Islamabad, Pakistan

Abstract

At LINAC Project PINSTECH, an electron gun test facility for indigenously developed 30 keV electron guns is developed to control and monitor various beam parameters by performing electron beam tests and diagnostics. After successful testing, electron gun is then integrated into 6 MeV standing wave linear accelerator. This paper presents the control system design and development for the facility.

INTRODUCTION

At LINAC Project PINSTECH, two RF standing wave linear accelerator prototypes are being developed. One is Medical linear accelerator (*Medical LINAC*) for use in medical applications such as cancer treatment, and the other is Industrial linear accelerator (*Industrial Linac*) for use in Non-Destructive Testing (NDT) applications. Electron source is one of the main components of an RF linear accelerator which provides charged particles to be accelerated by means of radio frequency [1]. Indigenously electron guns developed by Beam Transport and Diagnostics Group (BTDG) of LINAC Project are the electron sources for the linear accelerator prototypes at the project. Low energy 30 keV electron guns are designed and fabricated for integration into the 6 MeV linear accelerators. Characterization, testing and validation of electron gun parameters are important tasks that need to be properly done before integration into the accelerator [2]. For this purpose, Electron Source Operation and Characterization lab is developed at LINAC project PINSTECH that contains test facility. Next section describes overview of the experimental setup at the Lab.

OVERVIEW

The Lab is divided into two main areas: Experimental High Voltage (HV) Area and Control Room Low Voltage (LV) Area. Both of these areas are described in following subsections.

Experimental Area

Experimental Area consists of test bench and high voltage components. It consists of a low energy 30 keV electron gun along with beam diagnostic equipment integrated on a CF-63 based test bench. The designed electron gun is diode type based on thermionic emission which incorporates the modern dispenser cathode. The system is evacuated down to 1e-8 mbar with turbo molecular pump. Cold cathode ionization gauge IMG 300 is used for pressure measurement. A Faraday cup is integrated with transitional feed through to measure the beam current and a Ce doped YAG screen is used to measure the beam profile. The cathode of the gun is powered up with low voltage DC power supply, and beam extraction voltage is applied through high voltage DC power supply. Experimental area is remotely monitored and controlled from *Control Room* by means of a PLC-based control system. The schematic picture of experiment is shown in Fig. 1. Due to presence of high voltages, and for elimination of EMI, *Experimental Area* is completely caged, grounded and isolated.

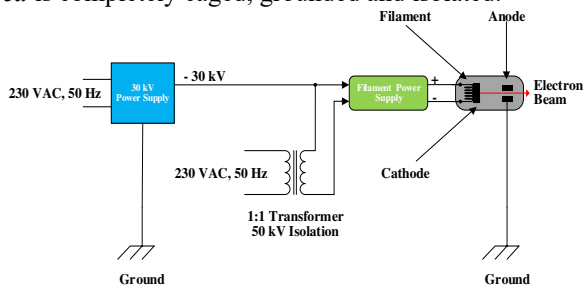


Figure 1: Schematic diagram of experiment.

Control Room Area

Control Room Area consists of two main components: *Operator Computer* and *Control Panel*. Operator computer consists of branded Dell i7 CPU, 24-inch LED monitor, keyboard, mouse and printer. The control panel consists of a C-type rail containing Siemens Simatic S7-300 PLC CPU, input, output and communication modules. The I/O modules allow support for digital inputs, digital outputs, analog inputs, analog outputs and RS-232/422/485 communications. The control panel can be divided into two parts: front panel and back panel. PLC controller along with its power supply, I/O and communication modules, and IPC477D touch panel are installed at the front panel. Circuit Breakers are installed at the back panel which provide protection in case of any short circuit or fault current.

CONTROL SYSTEM DESIGN

Control system acts as brain of the electron gun test facility. It enables operator to remotely monitor and control the complete system in a safe and interlocked environment to avoid any undesirable condition. In order to design the control system, hardware required was to have the following features: modular configuration for scalability, distributed configuration for separating high and low voltage area, Ethernet and serial interfaces for integration of commercial-off-the-shelf equipment, future availability for upgradation, knowledge and expertise for assistance. Keeping in view the above features, Siemens Simatic hardware and TIA-portal software packages were selected to design and develop the control system for electron gun test facility.

† nawazdanishali@gmail.com
* najm.control@gmail.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Logical Layout

The idea for development of the test bench control system is same as arrangement of experimental setup i.e. modular configuration divided into two areas: High Voltage (HV) and Low Voltage (LV). We need to monitor and control devices and interlocks inside the experimental area which is at high potential and also monitor and control electronic devices and interlocks from the control panel which is at the ground potential. LV area control module consists of main processing unit i.e. S7-300 CPU placed inside control panel situated in the Control Room Area, and HV area control module consists of slave unit i.e. ET-200M placed inside HV deck situated in the Experimental Area [3]. These two control system modules are connected via optical fiber to provide isolation. To program the PLCs, Ladder Logic programming language is mostly used. Logical layout of the control system is shown in Fig. 2.

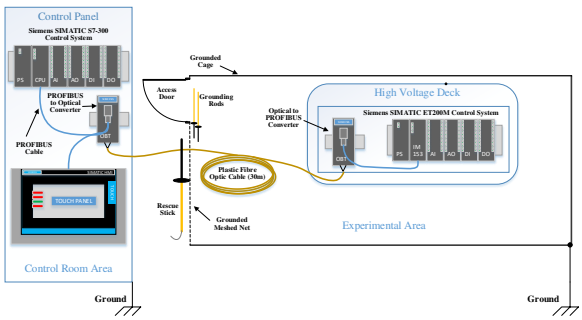


Figure 2: Logical layout of Control System.

Network Layout

CPU, slave module and touch panel need to communicate with one another for data transfer. This communication is achieved by means of a PROFINET-PROFIBUS network. Figure 3 shows the network layout configuration in Siemens TIA-Portal. ET200M based slave module is networked using PROFIBUS interface with the main processing unit S7-300. As PROFIBUS is a copper medium, it can carry EMI. Hence, PROFIBUS is converted to optical network and vice versa using Optical Bus Terminals (OBT) at both ends. ET200M module delivers/collects information to/from the main processing unit. PLC S7-300 CPU, IPC477D based HMI station, and operator computer are networked using ETHERNET interface as shown in Fig. 3.

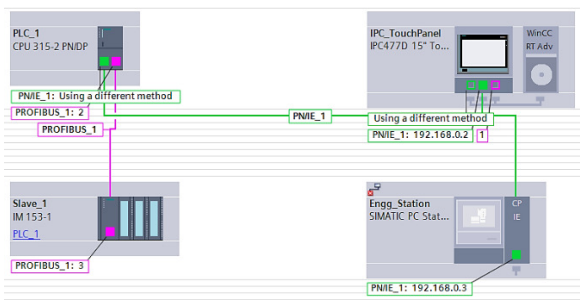


Figure 3: TIA-Portal Network layout.

Equipment

Several signals have been monitored and controlled from different instruments, power supplies, vacuum gauge controllers and faraday cup stepper motor. The brief explanation is given below:

Vacuum and Temperature

Vacuum of electron gun is measured from IMG-300 card of Agilent Technologies Vacuum Gauge Controller XGS 600. This card provide an analog signal corresponding to the vacuum and is acquired in the control system through an analog input module of S7-300. Desired/permissible vacuum window is programmed inside PLC for activating interlocks. Electron gun temperature is measured from K-type thermocouple through Phoenix transducer into the analog input module.

Filament Power Supply

TDK Lamba power supply is used for providing voltage to the filament. It is controlled and monitored through digital and analog input/outputs.

Pico ammeter

Keithley Pico ammeter is used to measure beam current. It contains serial interface (RS-485) for remote monitoring and control. RS-485 communication module is used to interface ammeter with PLC CPU.

High Voltage Power Supply

High Voltage is provided from Glassman High Voltage Power Supply. It is controlled and monitored through digital and analog inputs/outputs.

Solenoid Power Supply

Delta Electronika Power Supply is the solenoid power supply for the electron gun experimental setup. It is controlled from PLC via RS-232 interface.

Faraday Cup Motor

Faraday cup is moved back and forth with the help of stepper motor. Stepper motor driver receives pulse train, enable and director commands from PLC.

Interlocks

Several interlocks are incorporated for machine and personnel safety and protection with the help of PLC.

Human Machine Interface (HMI)

Two HMI stations are deployed in the test facility to remotely monitor and control the system. Both of these stations contain same HMI screens.

- PC based Human Machine Interface
- SIEMENS IPC477D Touch Panel based Human Machine Interface

Figure 4 shows layout diagram of the present deployed control system.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV019

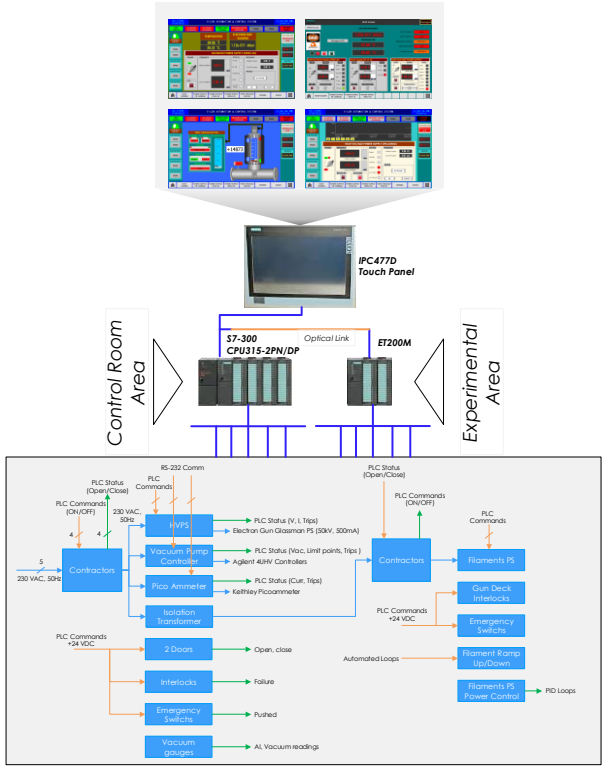


Figure 4: Control System Layout.

PLANNED UPGRADE

The existent PLC based control system is under process of upgradation to EPICS based control system. Operator interfaces currently developed in Siemens WinCC Advanced would be shifted to Phoebus Control System Studio and the existing Siemens PLCs will be interfaced with EPICS using s7nodave support module. Figure 5 shows layout of control system upgrade that will contain following additional hardware and software components:

High Voltage Pulse Modulator

Currently, Glassman high voltage DC power supply is used for extraction of electrons from the electron gun. This is a continuous DC mode power supply whereas the electron gun in RF linear accelerator is operated in pulsed mode. An indigenously developed 48 kV/110 A high voltage pulse modulator is developed for the facility. Control system for the modulator is already developed using S7-1200 PLC. It would be included in the system upgrade after interfacing with EPICS.

Magnet Power Supplies

Magnets are installed along the beamline for beam steering and focusing. Delta Electronika power supplies are installed for these magnets. The power supplies contain serial interfaces (RS-232/422/485) for remote interfacing. EPICS Asyn and Stream Device modules would be used to remotely monitor and control these power supplies via EPICS IOC.

Automated Software Loops

During system startup/shutdown, electron gun filament voltage is increased/decreased in steps to avoid filament damage. It is such a process which can be automated as it is based on a repetitive logic. We would automate this ramp up/down process of the electron gun filament using EPICS sequencer module. Sequencer implements finite state-machine using SNL (State Notation Language) which is a C-type language.

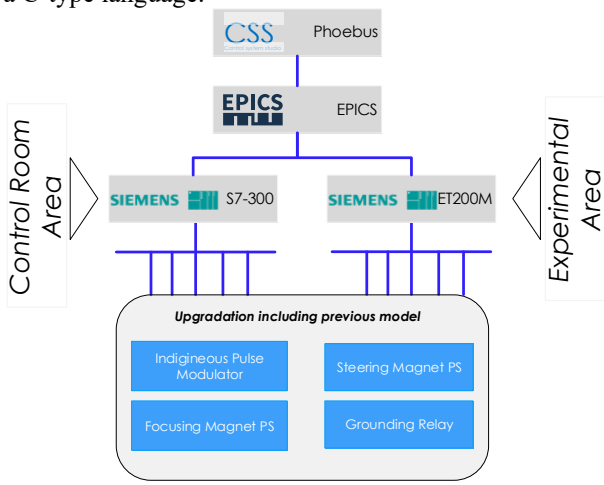


Figure 5: Control System Upgrade Layout.

EXPERIMENTAL SETUP

Experimental setup consists of Experimental Area and Control Room Area. Experimental Area is surrounded by Faraday cage which is grounded to eliminate EMI and provide protection from high voltages. Figure 6 shows the experimental area, Figure 7 shows the Faraday cage and Figure 8 shows the Control Room Area.

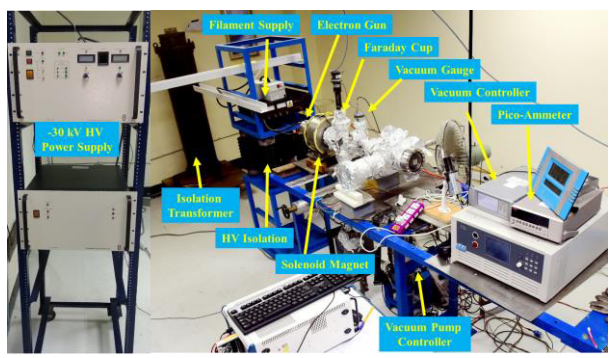


Figure 6: Experimental Area.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

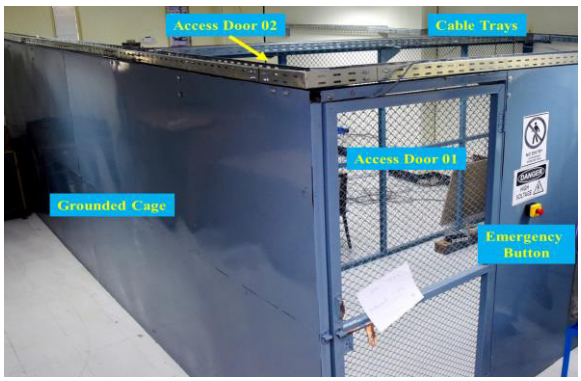


Figure 7: Faraday Cage.

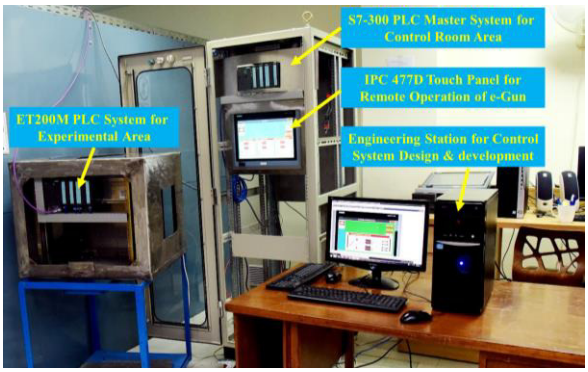


Figure 8: Control Room Area.

CONCLUSION

The automated electron gun test facility is efficiently providing infrastructure to test and characterize indigenously developed electron guns for integration into the RF linear accelerators. The planned upgrade of control system will further enhance the capabilities of the test stand for pulsed beam diagnostics and analysis.

ACKNOWLEDGEMENTS

The authors thank Beam Transport and Diagnostics Group (BTDG) of LINAC Project for their support that made this work possible.

REFERENCES

[1] K. Pepitone, and S. Doebert, “LEETHCI: The High Current Electron Source for the CLIC Drive Beam Injector”, in Proc. LINAC’16, East Lansing, MI, USA, Sep.2016, pp. 870-872. doi:10.18429/JACoW-LINAC2016-THPLR013

[2] C. Henkel, K. Flöttmann, W. Hillert, and V. Miltchev, “Characterization of an Electron Gun Test Setup Based on Multipacting”, in Proc. 10th Int. Particle Accelerator Conf. (IPAC'19), Melbourne, Australia, May 2019, pp. 1961-1963. doi:10.18429/JACoW-IPAC2019-TUPTS013

[3] Zhou, Zusheng, He, Dayong & Chi, Yunlong. (2013). “Electron gun system for NSC KIPT linac”, Chinese Physics C. 38. doi:10.1088/1674-1137/38/6/067006

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

AUTOMATIC RF AND ELECTRON GUN FILAMENT CONDITIONING SYSTEMS FOR 6 MeV LINAC

A. Majid†, D. A. Nawaz, N. U. Saqib*, F. Sher
LINAC Project, PINSTECH, Islamabad, Pakistan

Abstract

RF conditioning of vacuum windows and RF cavities is a necessary task for eliminating poor vacuum caused by outgassing and contamination. Also, startup and shutdown process of linear accelerator requires gradual increase and decrease of electron gun filament voltage to avoid damage to the filament. This paper presents an EPICS based multi-loop automatic RF conditioning system and Electron Gun filament conditioning system for Klystron based 6 MeV Linear Accelerator.

INTRODUCTION

Particle accelerators are a crucial instrument for scientific innovations and knowledge in all fields of research and engineering. As accelerators grow more complex, so do the demands placed on their control systems. The complexity of the control system hardware and software reflects the complexity of the machine. At the very same time, the machine must have simple access and operation, as well as a greater level of stability and adaptability.

Radio Frequency (RF) charged particle accelerators use accelerating RF cavities to accelerate electrons or ions to energies up to hundred mega electron volts. A particle source produces ions or electrons, which inject into the cavity with some minimum initial energy. A significant amount of RF power must be coupled into the accelerating cavity structure to transfer energy to the charged particles. High power klystron amplifiers are commonly employed as a RF power generation source for accelerators. In addition, waveguides and couplers are utilized to transmit RF power from the klystrons to the cavities. Injected RF power resonates with incoming particles, accelerating them to produce a higher energy, faster moving charged particle beam.

Several auxiliary systems, such as a water load and cooling system, vacuum maintenance system, control system and safety interlocks support the entire process. Maintaining a good quality vacuum in the LINAC cavity and connected components is critical for uninterrupted beam transport and smooth operation. When RF power is fed into the cavity during start-up and operation of the accelerator, the vacuum level degrades owing to poor manufacturing of the cavity's internal structures, RF breakdowns, outgassing, and arching. To cope with this issue, an auxiliary vacuum system is employed which consists of one or more series-connected pumps that continually generate a low-pressure zone inside the cavities. Radio frequency (RF) breakdown is a typical problem in accelerating structures and has been widely researched [1, 2, 3]. It is vital to avoid severe RF breakdown since it can harm accelerating structures and microwave devices irreparably. Breakdowns and multipacing restrict the power that the cavity can absorb during the conditioning process [4]. This phenomenon limits the RF structure from working in high-power mode and from producing full-energy beams. The frequency of breakdown is highly linked with the vacuum state and RF power intensity. Outgassing and sparking impact vacuum performance and restrict the RF field level that can be consistently attained.

Therefore, for safe operation, RF power is gradually injected into the cavity, ramping up in multiple stages from the lowest to the highest level while constantly monitoring the LINAC characteristic parameters such as vacuum level, temperature profile, arching, and other system interlocks [5]. The process of gradually "warming up" the LINAC cavity is known as RF conditioning. The amount of time required for conditioning, varies on the nature of cavity and might range from a few days to months.

This paper presents the design and development of a system for automated RF conditioning of electron linear accelerator cavities at the LINAC project PINSTECH. RF power conditioning scheme, hardware setup, logic design and software development techniques have been thoroughly addressed. The suggested structure is implemented with the Experimental Physics and Industrial Control System (EPICS), which is a collection of software components and tools used by developers to create distributed control systems for particle accelerators, big experiments, and massive telescopes [6]. In the last section, test setup developed for automated conditioning of electron gun filament has been discussed.

CONDITIONING STRATEGY

Numerous conditioning strategies have been employed at different accelerator facilities [7, 8, 9]. Linear accelerator prototypes (Medical and Industrial) at the LINAC project PINSTECH comprises of standing wave cavities structure, produces 06 MeV electron beam and runs at 2.5MW input RF power at resonance mode when the cavity is fully tuned. Electrons are injected by a 30keV electron gun and RF power is applied by a klystron that generates an oscillating RF signal of 50Hz frequency, 2.5 MW peak power and 5usec pulse width. The average power effectively delivered into the cavity is determined by three factors: peak amplitude, pulse width, and frequency.

$$P_{inj} = P_{peak} \cdot T_{on} \cdot f$$

Where P_{inj}, P_{peak}, T_{on}, f represent injected power, peak power, pulse width and frequency, respectively. These three parameters can be changed to enhance or reduce injected power.

† aaminahmajid@gmail.com
* najm.control@gmail.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

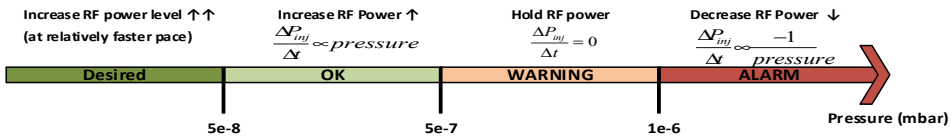


Figure 1: RF power conditioning strategy w.r.t vacuum dynamics of accelerator cavity.

At the beginning of the conditioning process, the lowest possible RF power is injected and its strength is increased gradually while constantly monitoring the vacuum quality. Magnitude and direction of power gradient $\frac{\Delta P_{inj}}{\Delta t}$ is adjusted based on vacuum dynamics. Pressure level data is a solid indication of breakdowns and outgassing phenomenon occurring inside the cavity. The desired operating range of pressure for power ramp up is 5×10^{-8} mbar or below. If the pressure rises up to 5.0×10^{-7} mbar, a high-pressure warning signal is generated and the RF power ramp-up process is halted. Consequently, pressure tends to settle down and vacuum quality improves. However, if the vacuum quality deteriorates further and pressure level exceeds 1×10^{-6} mbar, a vacuum interlock is triggered and RF power is ramped down instead, with a relatively greater step size. Figure 1 illustrates the relationship between RF power gradient with pressure level of cavity.

Power ramp-up scheme consists of three nested loops that correspond to the RF pulse parameters P_{peak} , T_{on} and f. These parameters act as main incrementing variables of respective loops. They are initialized at lower values and increased by a step size with each iteration of loop. Basic flow of the program is described in following steps:

1. Set Peak power, rep rate and pulse width of RF pulses to an initial value of 20kW, 0.7usec and 5Hz, respectively.

Innermost Loop (Increase RF Pulse peak power)

2. Increase RF pulse amplitude gradually while constantly monitoring the vacuum level. As power level rises, breakdowns begin to occur and vacuum level shows frequent surges. Depending on vacuum gradients and trends, adjust the step size and delay between two successive levels of RF power.
3. Condition to the maximum power that can be obtained without frequently activating the vacuum interlock.
4. After the pressure and breakdown rate have been reduced, increase the RF power to a higher level. Repeat this procedure until peak power reaches 800kW.

Middle Loop (Increase RF Pulse width)

5. Increase pulse width gradually up to 5 usec with step-size, $\Delta T_{on} = 0.2\,usec$. For each increased level of pulse width, repeat steps 2-4 until 1.6MW peak power has been achieved.

Outermost Loop (Increase RF Pulse rep rate)

6. Increase pulse rep rate gradually up to 50Hz with a step-size $\Delta f = 5Hz$. For each increased level of frequency, repeat steps 2-5 until RF pulse peak power has reached up to 2.5MW with 50Hz rep rate and 5usec pulse width.

HARDWARE DESIGN AND INSTRUMENTATION

The suggested RF conditioning methodology employs a variety of cutting-edge devices and instruments from diverse vendors. The following are the primary components employed in the conditioning setup: vacuum gauge controller, programmable logic controller (PLC), Inverted magnetron gauges, signal generator, peak power meter, oscilloscope, arc detector, delay generator, high speed RF interlock switch, temperature sensors, relays, safety interlocks, klystron and water-cooling system. The overall block diagram of the RF conditioning system is shown in Fig. 2.

RF signal is produced by signal generator, and passed to the klystron through an RF interlock switch. This switch is triggered by a gating sequence generated by the delay generator. As a result, the incoming RF pulse is transmitted only for a specific duration of time, obtaining the desired pulse width. This pulsed RF power is amplified by klystron and transferred to the cavity through waveguides.

During the RF conditioning process of high-power vacuum components such as cavities, couplers, windows, etc., it is necessary to monitor the vacuum quality, arcing, and forward and reflected RF-power levels. To monitor forward and reflected power, two directional couplers are connected to the wave-guide. Signals from directional couplers are measured with a high-speed oscilloscope that is interfaced with control software EPICS through TCP/IP. A pumping station consisting of a rotary and turbo pump linked in series maintains the vacuum level inside the cavity. These pumps operate continuously in order to attain the specified set point. Pumping speed is estimated based on pressure level feedback from vacuum gauges. Two IMG gauges are used to measure the pressure within the cavity at two different locations. A vacuum gauge controller is used to read gauge data and deliver it to the PLC through its analogue output.

Siemens S7-300 PLC system has been employed, as a field controller, to collect pressure readings and monitor auxiliary systems such as water load, temperature control, and arc detection. The PLC controller communicates withf Sensors/actuators and exchanges data with the primary control software built in EPICS. PLC is connected with arc detector, pressure switches, temperature sensors, and flow meter. In case of a malfunction or an abnormal circumstance, the PLC generates an interlock signal, which disables the RF switch.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

LOGIC DESIGN AND SOFTWARE DEVELOPMENT

Control system architecture is developed in the EPICS environment that also serves as an integration tool for all software components. EPICS StreamDevice support module is configured and installed to communicate with field instruments via TCP/IP or RS232/485 protocol. StreamDevice is a generic EPICS device support for devices with a "byte stream" based communication interface. For each device that needs to be interfaced with control software, a soft IOC (input-output controller) is developed in EPICS. These IOCs act as servers, collecting data from the attached instrument and broadcasting it to many clients through the channel access networking protocol.

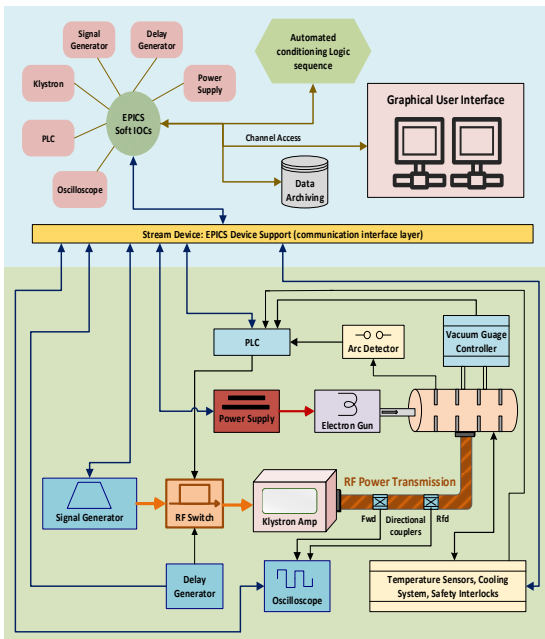


Figure 2: Hardware setup and instrumentation.

For conditioning tasks, soft IOCs have been developed and configured that interface with their respective equipment. Examples include klystron, signal generator, e-gun power supply, delay generator and PLC, allowing their complete remote access. Each IOC has a dedicated database that stores information in the form of records or process variables (PVs). PVs are configured and linked with the real-system field parameters. IOCs can communicate with other IOCs, as well as with databases and front-end applications. All process variables are accessible anywhere on the network via channel access protocol.

The EPICS sequencer module is used to program the algorithm for the automated conditioning system in state notation language (SNL). This application operates in the background and exchanges data with the relevant IOCS and databases. It reads the most recent values of process variables and makes decisions according to control logic. Front-end application often known as Graphical User Interface (GUI) is developed in Control System Studio-Phoebus. An interactive graphical user interface has been created, allowing the user to modify and monitor associated process variables and their trends. Operators can connect and communicate with any instrument from any remote location using the front-end application. To perform RF conditioning, operator can send start/stop requests and set values of related parameters such as initial power, final power, step size and time delay etc.

These values are sent to the sequencer program that evaluates the next decision and updates relevant PVs in the database. IOCs serve as a link between database and field instruments for updated information exchange. The sequencer program is written in the form of states, and its flow chart is depicted in Fig. 3. It has five states: ***Start, Vacuum Monitor, Increment Decrement*** and ***Fault.***

program begins with the *'Start'* state, at the request of operator's start command, initializing all variables: Initial and final RF pulse frequency, rep rate, amplitude and the step sizes denoted by f_0, f-max, T_0, T_{on}-max, P_0, P-max, Δf, ΔT_{on} and ΔP, respectively. If all the interlocks are cleared and system is in ready mode, program advances to the next state, *Vacuum Monitor.*

In the *Vacuum Monitor* state, program waits for a set period Δt (default: 2 min) and monitors pressure data with 1 Hz frequency, making the next decision based on the vacuum quality. This decision can be further divided into three segments: a) If pressure level remains in normal range for the entire two minutes, it advances to *Increment* state. b) If pressure increases to the warning level, program returns to the same *Vacuum monitor* state and waits for the pressure to settle down. c) If pressure continues to rise and crosses the alarm limit, program immediately switches to the *Decrement* state.

In the *Increment* state, program steps up the RF power level in three nested incrementing loops. It achieves this, by iteratively increasing pulse peak amplitude, pulse width or frequency based on the preceding level in the loop and step size. After stepping up RF power, it returns to *Vacuum Monitor* state.

In the *Decrement* state, program steps down the power by decreasing RF pulse amplitude. After reducing the RF power level, it pauses for a defined period before returning to the *Vacuum Monitor* state.

Meanwhile, if system interlock occurs, program enters

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

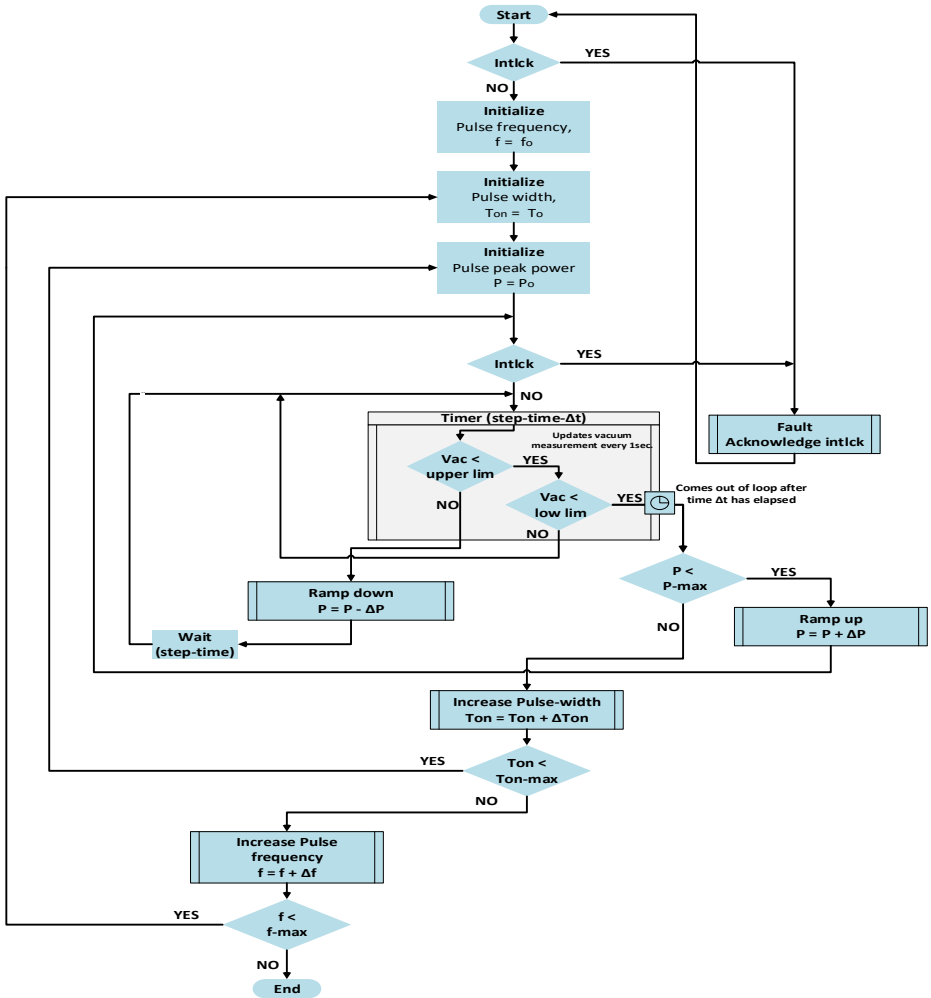


Figure 3: Flow chart of conditioning algorithm.

the *Fault* state and turns off RF power. Once the fault has been recovered and acknowledged, the conditioning process can be restarted only if the operator sends the start instruction. When the maximum required level of RF power is attained, the conditioning process is terminated, and the program transmits a complete signal to the database and GUI. The program can be paused at any time during operation through GUI. Furthermore, all associated parameters such as power gradient, step sizes, time delays, and limitations may be adjusted at any moment for a more flexible operation and conditioning procedure.

ELECTRON GUN LOW VOLTAGE CONDITIONING

A high voltage DC electron gun generates the electron beam that is injected into LINAC. The gun filament is composed of tungsten and is housed in a chamber filled with barium oxide. A tungsten disc is positioned at the end of the chamber and is primarily responsible for electron emission. A low voltage and high current (12V, 2.5Amp) is supplied to the filament that heats the tungsten disc indirectly. As a result, when the disc attains a breakdown temperature of 1200°C, it begins releasing electrons with a few electron volts of energy. These electrons are then excited by a 30 KV pulsed power supply and kicked into the accelerating cavities for further gain in energy and speed.

During the low voltage (LV) excitation of electron gun filament, a sudden rise in temperature might cause thermal breakdowns and damage the electrodes. Therefore, for the safe beam on operation, low voltage applied to the gun filament should be increased gradually from 0 to 10V to maintain a minimum temperature gradient. This is referred to as LV conditioning of electron gun filament. The conditioning process eliminates contaminants on the insulating ceramic and the electrode surface, avoids poisoning of electrodes and improves breakdown threshold. During LV conditioning a vacuum level of 1×10^{-8} mbar is maintained to transport electron beam smoothly and avoid electric discharges. Voltage ramp up strategy must be in accordance with the vacuum quality.

A test setup has been developed for automatic LV conditioning and ramp up of electron gun filament voltage. Gun filament is powered up by DC power supply, which is serially interfaced with EPICS soft IOC. A serial to optical converter has been used as an intermediate link to provide

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

isolation from power supply. All parameters of power supply: set voltage, actual voltage and actual current are linked with IOC. Their values are also accessible from database and GUI. A sequencer program is developed for automatically ramping up the power supply voltage with respect to the vacuum level. A single loop program increases, holds or decreases gun voltage according to pressure level of cavity. Vacuum warning and alarm limits are adjustable from the GUI. Program continuously monitors interlocks and pauses the ramp-up process in case of any alarm or fault. This is to prevent excessive discharges during the conditioning.

RESULTS

Automatic RF conditioning system has been developed, installed and tested at LINAC Project, PINSTECH. Numerous experiments have been conducted using the developed software and performance is satisfactory. This setup facilitates users for safe and reliable conditioning of accelerator cavities with minimum human intervention.

Figure 4 depicts a specimen of data during one of the RF Power conditioning trials. It shows 150-minute span of the process where injected power and pressure measurements are represented by red and green curves, respectively. RF power was systematically increased by the control software. Discontinuities in graph (around: 11.30 and 12:00) reflect the RF power cut-down events due to vacuum spikes and system fault. In the beginning [11.00-11:30], RF power was increased with larger step, but for the remaining period power gradient was decreased due to small margin in pressure level and warning limit.

Figure 5 illustrates a snapshot from electron gun LV conditioning process that was being controlled by the developed software. Electron gun filament voltage and cavity pressure are plotted by red and blue colors, respectively. Their relative trend in graph can be divided into three categories: First segment [12:36-12:50] represents 'Increment' state where vacuum quality is good and filament voltage is increased smoothly with constant rate, Second segment [12:50-13:05] shows 'Hold' state where vacuum is in warning range, and program holds the filament voltage. Third and the last segment shows voltage 'Decrement' state, where pressure crosses alarm limit at 13:07 and voltage is decreased quickly.

Figure 6 displays Operator Interfaces (OPI) for RF conditioning and e-gun conditioning systems. GUIs are user-friendly and facilitate operator with the provision of adjusting variables (limits/step sizes), acknowledging alarms as well as initiating, monitoring or ceasing the conditioning process.

CONCLUSION

An optimal scheme for RF power conditioning of standing wave linear accelerators is proposed and discussed. Aim of this work was to develop an automated system to perform recommended conditioning strategy in a reliable

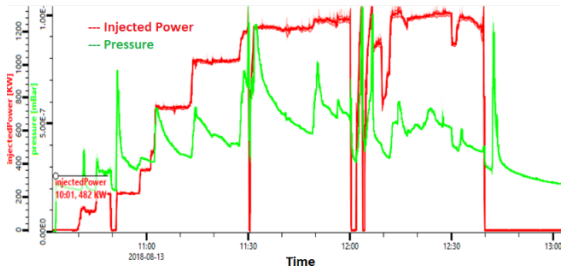


Figure 4: RF Power conditioning Process.

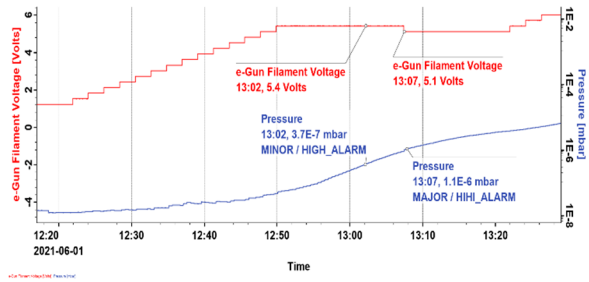


Figure 5: e-Gun LV conditioning.

Figure 6: OPIs for RF & e-gun conditioning.

and efficient manner with minimum surveillance. The proposed system has been successfully implemented in EPICS platform and PLC has been used as main controller for interfacing of hardware electronics. Both software and hardware design strategies have been discussed. All components and instruments used for this project are easily available commercially. System's parameters are accessible and reconfigurable based on the requirement of end-user. A similar setup for LV conditioning of electron gun filament has also been developed. This system has been tested and installed at LINAC Project, PINSTECH. It has been effectively used during several experimental runs and its performance is well in-line with the desired requirements.

REFERENCES

[1] V. Dolgashev, S. Tantawi, Y. Higashi, and B. Spataro, "Geometric dependence of radio-frequency breakdown in normal conducting accelerating structures", Appl. Phys. Lett. 97, 171501 (2010).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV020

[2] F. Wang, C. Adolphsen, and C. Nantista, "Study of radio frequency breakdown in pressurized L-band waveguide for the International Linear Collider",Appl. Phys. Lett. 103, 104106 (2013).

[3] T. Garvey, and W.-D. Möller, "The RF power coupler development programme at LAL-Orsay and DESY-Hamburg for TESLA and the European X-FEL", International Congress on Optics and Opto-Eletronics, Warsaw, Poland (2005).

[4] S. Papadopoulos, J. M. Balula, F. Gerigk, J-M. Giguet, J. Hansen, A. Michet, S. Ramberger, N. Thaus, R. Wegner, "Experience with the conditioning of LINAC-4 RF cavities", in *Proc. LINAC2016*, East Lansing, MI, USA, Sep. 2016, pp. 985-988. doi:10.18429/JACoW-LINAC2016-THPLR060

[5] Q. Chen, T. Hu, B. Qin, Y. Xiong, and K. Fan, "RF conditioning and breakdown analysis of a traveling wave LINAC with collinear load cells", Phys. Rev. Accel. and Beams 21, 042003 (2018).

[6] EPICS, https://epics-controls.org

[7] T. Ohshima, H. Suzuki, H. Ego, M. Hara, N. Hosoda, Y. Kawashima, Y. Ohashi, F. Wang, and H. Yonehara, "Conditioning of RF cavities and RF input couplers for SPring-8", in *Proc. EPAC'96*, Sitges, Spain, Jun. 1996, paper WEP080L, pp. 2047-2049.

[8] S. M. S. Hasan, Y. W. Kang, and M. K. Howlader, "Development of an RF Conditioning System for Charged-Particle Accelerators", IEEE Transactions on Instrementation and Measurment, vol. 57, no. 4, (2008).

[9] H. Li, C. Maiano, E. Conejero, and R. Zeng, "An optimal procedure for coupler conditioning for ESS superconducting LINAC", in *Proc. 18th International Conference on RF Superconductivity (SRF2017)*, Lanzhou, China, Jul. 2017, pp. 103-107. doi:10.18429/JACoW-SRF2017-MOPB020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV025

CONTROL SYSTEM OF UPGRADED HIGH VOLTAGE FOR ATLAS TILE CALORIMETER

F. Martins[1,*,†], F. Cuim[1], G. Evans[1,2], R. Fernandez[1], L. Gurriana[1], A. Gomes[1,2], J. Soares Augusto[2,3]
[1] Laboratory of Instrumentation and Experimental Particle Physics (LIP), Lisbon, Portugal
[2] Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal
[3] Inesc-ID, Lisbon, Portugal

Abstract

The preparation of the upgrade of the ATLAS electronics for the High Luminosity LHC is in full swing. The Tile Calorimeter is preparing the upgrade of its readout electronics and power distribution systems. One of such systems is the High Voltage (HV) regulation and distribution system. The new system is based on HVRemote boards mounted in crates located at the counting room. The HV will be delivered to the on-detector electronics using 100 m long cables. The crates will be equipped with a system-on-chip that will be responsible for the control and monitoring of the HV boards. The control of the HVRemote and its dedicated HVSupply boards is done by means of a serial peripheral interface bus. A SCADA component is under development to communicate with and supervise the crates and boards, and to integrate the HV system in the control system of the detector. The control system will be able to send notifications to the operators when the monitored values are out of range, archive the monitored data and if required, perform automated actions.

INTRODUCTION

Located in the central region of the ATLAS detector [1], the Tile Calorimeter (TileCal) [2] is composed of iron plates interpolated with plastic scintillating tiles that work as active material. The light resulting from the interaction of the particles with the plastic tiles is guided by wavelength shifting optical fibers to the photomultipliers tubes (PMT). The signal generated by the PMT is then processed by the on-detector and off-detector data acquisition electronics. TileCal uses approximately 10^4 PMTs divided over the 256 modules that compose the four TileCal operational barrels (EBA, EBC, LBA and LBC). The operation of TileCal requires a high voltage system able to regulate and monitor the HV set to each individual PMT with a precision within 0.5 V rms. For the High Luminosity Large Hadron Collider (HL-LHC) a new high voltage system is under development [3, 4] for TileCal, along with other hardware upgrades [5, 6]. It will consist of HV source boards, regulation and monitoring boards housed in crates located in the off-detector area, with the HV being delivered to each PMT by 100 m long cables. With the regulation boards located off-detector, they are not required to be radiation hard and they have also the advantage of easier access for repairs which can be executed during data-taking periods.

The Detector Control System (DCS) of TileCal is responsible for the supervision and control of the TileCal electronics and infrastructure, continuously monitoring temperatures, voltages and currents [7].

HIGH VOLTAGE BOARDS

HVSupply

The HVSupply board is responsible for providing the direct current (DC) HV and the three low DC voltage levels, (12 V, −12 V, 3.3 V) required by the HVRemote board to operate. The HV will be supplied by two commercial DC/DC converters (Hamamatsu C12446-12) with a combined maximum output current of 20 mA and adjustable voltages up to −1000 V. The supervision will include the monitoring of the supplied voltages and respective current consumption as well as the readings of on-board temperature probes and on-chip temperature. The digital control and monitoring is achieved by using a dedicated Serial Peripheral Interface (SPI) bus. An analog switch was also implemented in the board which will allow the HV DC/DC converters to be switched off when a hardware interlock signal is removed. The interlock signal can either be interrupted by the TileCal DCS or by the Detector Safety System (DSS).

HVRemote

The HVRemote board is composed of 48 HV channels for regulating and monitoring the individual channels voltages in the range from −500 V to −950 V. Enabling or disabling the output of the HV channel is available by software in groups of 4 channels. However it is possible to disable the output of any of the HV channels by hardware (by means of a jumper) allowing, for example, to isolate it from a short circuit in the long cable or in the HVBus board (located on-detector). Temperature probes are available to monitor the board temperature and assess the cooling conditions. Each of the boards will have an unique serial number allowing the crate controller and the Supervisory Control and Data Acquisition (SCADA) control system to identify each individual HVRemote board. The communication between the on-board circuits and the crate control is done via a dedicated SPI bus. Figure 1 shows a simplified diagram of the HVRemote board control.

* Corresponding author fmartins@lip.pt
† Copyright 2021 CERN for the benefit of the ATLAS Collaboration. CC-BY-4.0 license.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

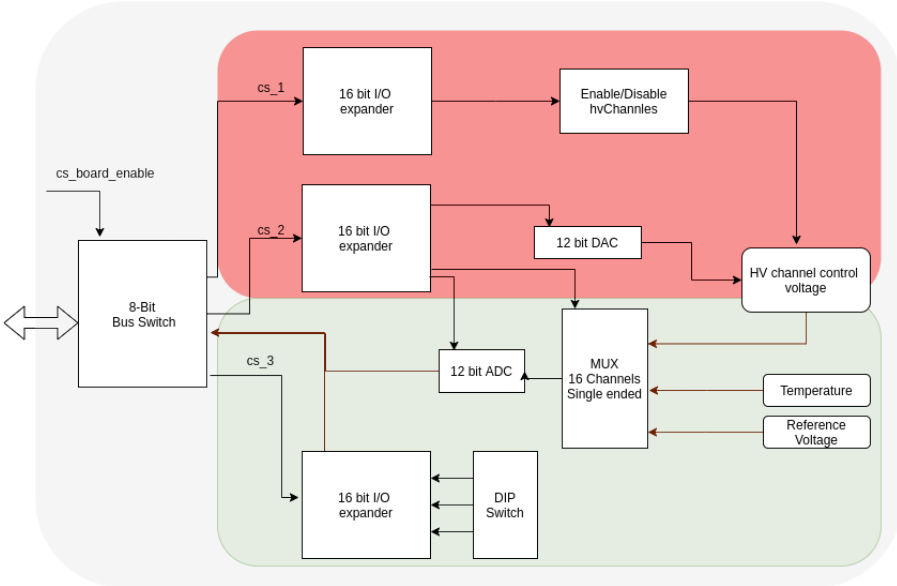


Figure 1: Simplified block diagram of the HVRemote control. The area of the diagram with the pink background represents the control of the HV and the one with the green background represents the monitoring path.The enabling of the Switch allows a specific board to receive the SPI commands from the controller.

HV Bus Board

The HV-bus boards root the HV delivered by the 100 m long cable to the designated PMT. Each board is equipped for 12 PMTs and 3 (in EBA/EBC) and 4 (in LBA/LBC) boards are needed per TileCal module. The HVbus is the only element of the HV distribution and regulation system which is located on-detector but since it does not have active elements (only resistors) there is no specific radiation requirement.

CRATE CONTROL

The control of the first HVRemote prototype was based in the Tibbo board and it was mounted directly on the HVRemote board [3]. Taking this approach, the final system would require the availability of more than 500 ethernet sockets and respective network cables, which would be a challenge due to space restrictions on the racks at the electronics rooms. At the end it would also be a complex system to maintain since a large number of Internet Protocol (IP) addresses would had to be kept and correctly mapped inside of DCS. To avoid dependencies on such constrains as well as to make the maintenance of the HV system easier and at the same time allowing the upgrade of the controller board if required, it was chosen to have a single controller installed on the crate. In this new design, the controller will be used to control both the HVRemote and the HVSupply, each with its dedicated SPI bus. Thus it was decided to use a System-on-chip (SoC) for the final crate control.

Interface Board

The interface board, that will be connected in the crates back-plane will hold a ZYBO board [8] and two port expanders. The ZYBO board is equipped with a Zynq-7000 SoC featuring a dual-core *ARM* processor and a Field Programmable Gate Array (FPGA) logic from *Xilinx*. The board is equipped with an EEPROM programmed with an unique Media Access Control (MAC) address, a built-in RJ45 connector for ethernet, Pmod connectors, among other features. The General Purpose Input and Output (GPIO) and SPI signals were assigned using *Vivado* and can be accessed at the assigned Pmod ports/pins. The boot of system was created with the *PetaLinux* tool integrating the design developed with *Vivado*. Relative to the Linux distributions, we are currently testing with *CentOS* 7 and *Ubuntu*. The root filesystem (rootfs) and boot are executed from the local Secure Digital (SD) card. There are two port expanders located on the interface board that will be used to select one board at a time, the boards on each SPI bus (each bus will be operated independently from the other). For this matter, a set of GPIO signals will be used to enable the assigned bus port expander, instead of the standard SPI chip select.

Software

The software running on the SoC (or in the Raspberry Pi used in the tests) operates as SPI controller [9]. The actual software for the board control has been written in Python and it has been used a Raspberry Pi 4 to develop it. Libraries implementing the required SPI instructions have been prepared to operate the control chips within each board. For example, setting I/O pins from the port expanders with a specific configuration. The communication between the DCS and the controller is (currently) a simple home-made protocol over TCP/IP allowing the commands or queries to be sent from the DCS to the controller and replies flow in the opposite direction, see Fig. 2. When the server is started it initiates the SPI bus and creates a socket which listens and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV025

waits for commands or queries. At the restart of the server after a crash or un-programmed reboot of the SPI controller, the server does not initialize the boards, avoiding this way losing the previous configurations. From the DCS side, there are two types of requests; automatic queries (usually readings) which are updated regularly, and the commands which are initiated by the user and have priority over the automatic queries. Queries and commands handling are implemented in the software using a "to-do" list, where a process takes the command/query from the top of the list and sends it to the server to be handled. When the handling of the query or command is complete, the process takes the next item from the list and the process repeats itself. When the list is empty, the software updates the list with the group of automatic queries.

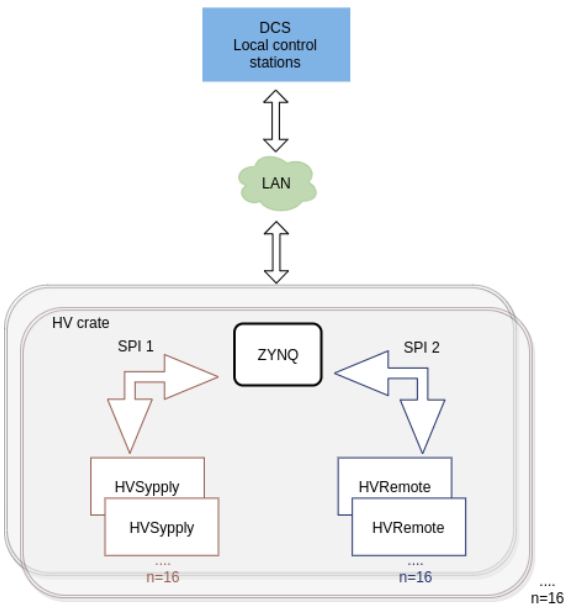


Figure 2: Architecture of the new HV system control tree.

INTEGRATION WITH TILECAL DCS

The slow control system of TileCal is based on the SCADA *WinCC OA* [10] and uses a set of software packages that ease the coherent integration of the hardware and its respective control component into the DCS. TileCal DCS is a distributed system comprising (currently) six individual SCADA systems, occupying two layers of the ATLAS DCS back-end hierarchy (see Fig. 3) taking different roles depending on the layer and type. The local control stations are the systems responsible for handling the hardware systems such as low voltages and high voltage [7].

The SCADA component for the HV Crates is being developed and under initial tests. Data point structures that represent the crate and respective boards were prepared, containing temperature probes, voltages and calibration values. The data arriving from the HVSupply and HVRemote are decoded and mapped to the respective data point element. It was also prepared a set of basic Graphical User Interfaces (GUI) which display the data located in the data points elements and execute commands such as switching on/off HV channels (see Fig. 4), board resets (initialization), among others.

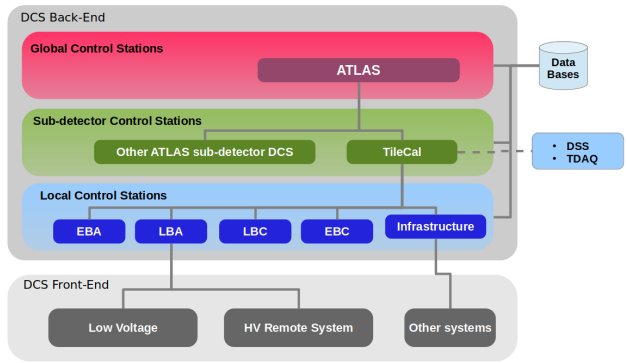


Figure 3: TileCal DCS hierarchy. Image adapted from [7].

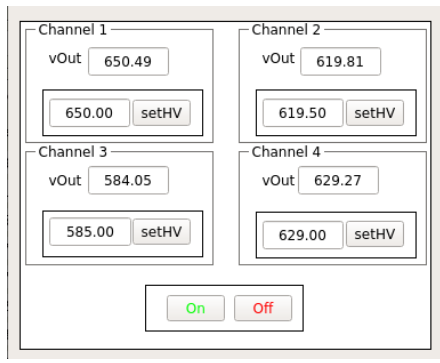


Figure 4: DCS test user interface. The measured HV is shown in the display "vOut" for a group of four channels. It shows the HV for each of the channels. It is also possible to enable or disable the HV for all four channels.

The archiving of the monitored data to the Oracle database is performed according to a set of rules or conditions (smoothing). The smoothing reduces the amount of data stored without losing important data, therefore making the archiving more efficient. For the HVRemote Channels, notifications are available in the form of alerts, which are triggered when the offset between the requested HV and the monitored HV is higher than 4 V (the value can be adjusted channel by channel). Notifications to experts such as emails or a short message service can also be configured, for example in the case of HVSupply failure. Actions can be triggered by alarms conditions as well, for example, in the case of high board temperature, the DCS should power down the HVSupply and HVRemote as a preventive measure.

Figure 5 shows the monitored HV by DCS from source 1 of an HVSupply board, without smoothing.

CONCLUSIONS AND FUTURE WORK

A new control is being prepared for the HV remote system which will consist on control software running on the crates and supervision software which is part of the TileCal DCS. As for the next steps the crate control software will be ported

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

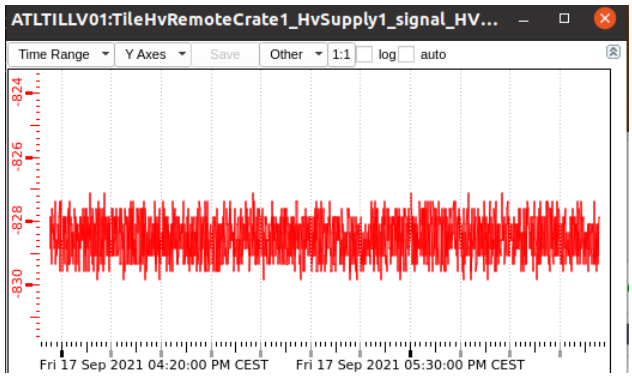


Figure 5: Plot of the output voltage (without smoothing) of one HV source located at the HVSupply during a test beam period with realistic data taking conditions.

to the ZYNQ controller and fully tested with the interface board and the other componentes of the system.

ACKNOWLEDGEMENTS

This work is funded in part by the "Fundação para a Ciência e a Tecnologia", Portugal, under the project "Upgrade da Experiência ATLAS", CERN/FIS-PAR/0033/2019.

REFERENCES

[1] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", *J. Instrum.*, vol. 3, p. S08003, 2008. 10.1088/1748-0221/3/08/s08003

[2] ATLAS TileCal Collaboration, "Tile Calorimeter Technical Design Report", ATLAS Internal Note, CERN/LHCC/96-42, 1996. https://cds.cern.ch/record/331062

[3] F. Martins, G.G. Evans, A. Gomes, L. Gurriana, A. Maio, C. Rato, J.M. Sabino, L. Seabra, and J.A. Soares Augusto, "Control System for Atlas Tilecal HVremote Boards", in *Proc. ICALEPCS2017*, Barcelona, Spain, Oct. 2017. doi:10.18429/JACoW-ICALEPCS2017-THPHA069

[4] A. Da Silva Gomes, J.A Soares, F. Cuim, G. Evans, R. Fernandez, L. Gurriana, and F. Martins, "Upgrade of the ATLAS Tile Calorimeter High Voltage System", in *Proc. of Topical Workshop on Electronics for Particle Physics — PoS(TWEPP2019)*, vol 370, p. 062, 2020. doi:10.22323/1.370.0062

[5] ATLAS Collaboration, "Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter", Sep 2017. https://cds.cern.ch/record/2285583

[6] E.V. Santurio, "Upgrade of Tile Calorimeter of the ATLAS Detector for the High Luminosity LHC", *J. Phys. Conf. Ser.*, vol. 928, p. 12024, 2017. doi:10.1088/1742-6596/928/1/012024

[7] F. Martins, "The ATLAS Tile calorimeter DCS for run 2," *IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*, pp. 1-5, 2016. doi:10.1109/NSSMIC.2016.8069837

[8] Dililent Zybo: https://digilent.com/reference/programmable-logic/zybo/start

[9] Open Source Hardware Association: A Resolution to Redefine SPI Signal Names, https://www.oshwa.org/a-resolution-to-redefine-spi-signal-names/

[10] Siemens SIMATIC SCADA Systems: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada.html

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV027

EPICS DAQ SYSTEM OF BEAM POSITION MONITOR AT THE KOMAC LINAC AND BEAMLINES

Young-Gi Song†, Jae-Ha Kim, Sung-Yun Cho
Korea Multi-purpose Accelerator Complex, Korea Atomic Energy Research Institute, Korea

Abstract

The KOMAC facility consists of low-energy component, including a 50-keV ion source, a low energy beam transport (LEBT), a 3-MeV radio-frequency quadrupole (RFQ), and a 20-MeV drift tube linac (DTL), as well as high-energy components, including seven DTL tanks for the 100-MeV proton beam [1]. The KOMAC has been operating 20-MeV and 100-MeV proton beam lines to provide proton beams for various applications. Approximately 20 stripline beam position monitors (BPMs) have been installed in KOMAC linac and beamlines. A data-acquisition (DAQ) system has been developed with various platforms in order to monitor beam position signals from linac and beamlines. This paper describes the hardware and software system and test results.

INTRODUCTION

Ten stripline BPMs and nine stripline BPMs were installed in the 350MHz pulse KOMAC Linac and beamline, respectively. Figure 1 shows the BPM installed in Linac and beamlines.

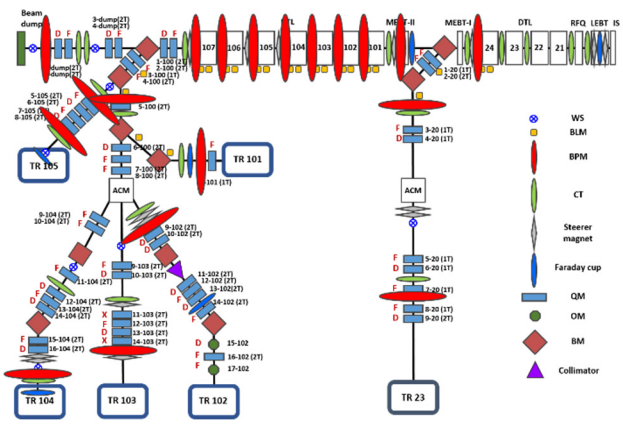


Figure 1: Layout of KOMAC Linac and Beamline.

In addition to the transverse beam position, the BPM is used to measure the beam phase for energy calculation based on flight time measurement. VME-based board was adopted as a major platform for high-performance subsystems. The MVME3100 CPU board has been adopted as a standard control system and is used in beam diagnosis, timing system, and LLRF control systems [2][3].

The DAQ system for BPM measures beam position and phase in pulse mode through IQ-based RF signal measurement. It used a single type of electronic product for all BPMs so that the design of the front electronic device is applied to all type of single-board computers through minor modifications. Table 1 summarizes the main specification of the Linac BPM and beamline BPM. The fabricated BPMs are shown in Fig. 2.

Table 1: Design Parameters of the BPMs

Type	Linac BPM	Beamline BPM
Electrode aperture	20 mm	100 mm
Electrode thickness	2 mm	2 mm
Electrode angle	60 deg.	456 deg.
Electrode length	25 mm	70 mm
Electrode gap	3.5 mm	15 mm
Feedthrough	SMA	SMA
Signal frequency	350/700 MHz	350 MHz

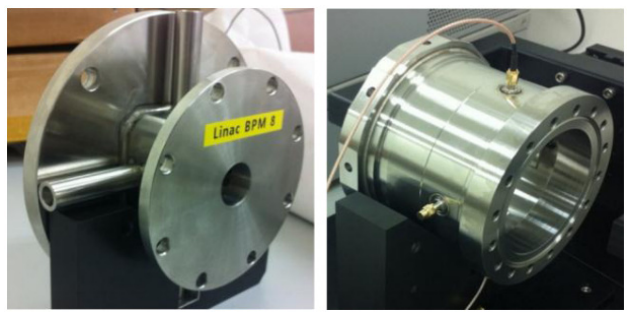


Figure 2: Linac BPM (left) and beamline BPM (right).

The BPM frontend that receives, levels down, and filters BPM signals includes analog parts of some sensitive electrical devices. The signal that has passed the frontend is sampled by the digitizer board and accumulated in memory. The sampled data extracts valid values such as beam position, beam phase, and beam current. KOMAC BPM DAQ has been upgraded to be integrated using EPICS [4] after collecting and processing BPM signals.

SYSTEM CONCEPT

As part of the available electronics platform and electronics standardization strategy, VME has been adopted and used as a major device for KOMAC's high-performance electronics, including BPM, timing and LLRF systems. The main reasons for adopting VME are excellent performance, durability, and cost reduction. In particular, by unifying the platforms of various control system, short development processes and simple maintenance become the biggest advantages.

Figure 3 shows a schematic diagram of the BPM system manufactured by KOMAC for accuracy and calibra-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV027

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

tion measurement. The original plan was to measure the beam phase and position using a log ratio BPM electronic device. However, only the beam phase of the BPM is currently being measured. The upgrade plan is to directly measure the four signals of the BPM electrodes and simultaneously generate beam phase, beam position, and beam current from four signals.

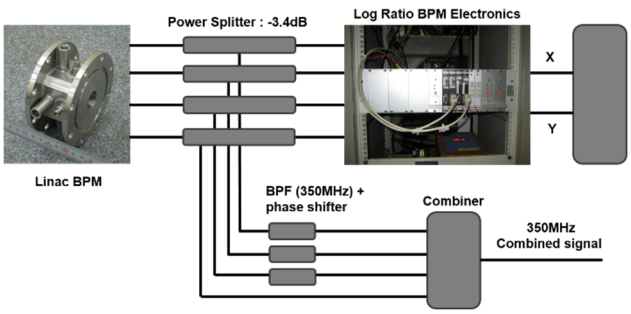


Figure 3: Original plan for beam phase and position.

The induced voltage of the BPM is transmitted to the DAQ system installed in the KOMAC klystron gallery by four coaxial cables. The DAQ contains several electronic modules, such as digitizer board for processing analog signals and VME CPU board with EPICS middleware installed on vxWorks operating system.

The accuracy and calibration of the BPM were performed on the test stand as shown in Fig. 4. After checking the signal processing results of BPM's test stand and DAQ, the calibration value is obtained by measuring the BPM position mapping data using the actual BPM signal line in the tunnel.

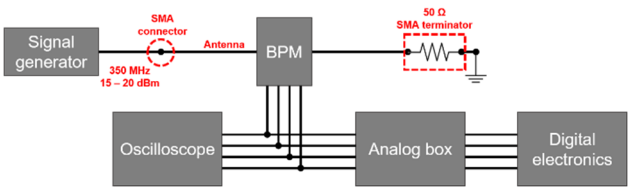


Figure 4: BPM test stand for accuracy and calibration measurement.

A wire containing RF power is passed through the BPM instead of the actual beam. The ADC board measures the X and Y output voltages of the BPM. The X and Y voltages sampled by the ADC are processed in driver support of EPICS IOC and serviced using Channel Access (CA).

BPM DAQ SYSTEM

The data acquisition board consists of a Linac BPM and beamline BPM. The MVME3100 baseboard and PENTEK7142 board for beam signal acquisition have been adopted as Linac BPM as shown in Fig. 5. For beamline BPM, Libera spark HL products have been adopted [5]. PENTEK7142 is a PMC type board with a 14-bit, 125MHz/s/channel sampling function for four independent analog inputs with Xilinx Virtex-4 FPGA. At the KOMAC facility, 10 DAQ boards for Linac BPM and 9 DAQ boards for beamline BPM are installed. Since the BPM DAQ must be synchronized with the beam timing, all DAQ system for beam diagnosis equipment must be synchronized with the KOMAC timing system.

Figure 5: MVME3100 and PENTEK7142 for Linac BPM.

One of the reasons for choosing the MVME3100 board is to match the PCI bus transmission rate with the PENTEK board. Table 2 describes the PCI bus specifications. The PENKEK board supports a PCI bus transmission rate of 64 Bits/66 MHz. The PCI bus of the MVME3100 may select 32bits and 64bits of bus width at 66 MHz.

Table 2: Specifications of PCI Bus

Type	Clock	Bus Width	Data Rate
PCI	33MHz	32bits	132MB/s
		64bits	265MB/s
PCI or PCI-X	66MHz	32bits	264MB/s
		64bits	528MB/s
PCI-X	133MHz	32bits	532MB/s
		64bits	1064MB/s

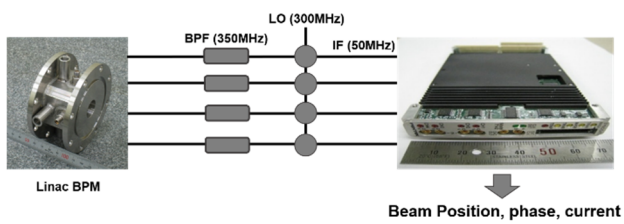


Figure 6: Schematic diagram of BPM DAQ System for measuring beam phase, position, and current.

The analog frontend consists of RF components including a mixer, 4-way splitter, 350MHz band pass filter, low pass filters. The 350MHz beam signal is mixed with the 300MHz LO signal for signal conversion and is down-converted to 50MHz IF signal in the analog frontend. The IF signal is sampled by the PMC ADC board. Figure 6 shows BPM DAQ System for measuring beam phase, position, and current with a 350MHz signal generator and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV027

a DAQ digitizer board used in the BPM test stand. The DAQ system for the Linac BPM is classified into two parts: MVME3100 baseboard and PMC digitizer board. In PMC, data is sampled using ADC and the sample IQ data is stacked in the FIFO memory. When the stored data reaches a specified memory size, an interrupt is generated and the data is transferred to the baseboard through a PCI bus. The data block transmission between the baseboard and the PMC digitizer board uses the DMA method. DMA uses burst address mode, which repeats read/write indefinitely from start to end. The burst address mode reduces the counter by 1 and proceeds until it reaches zero, and when it is done, it informs the CPU using DMA_INT as shown in Fig. 7.

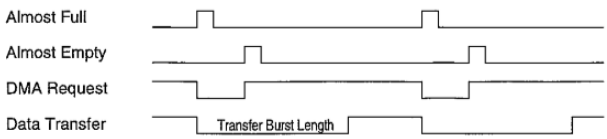


Figure 7: Transfer the data block between two-hardware devices.

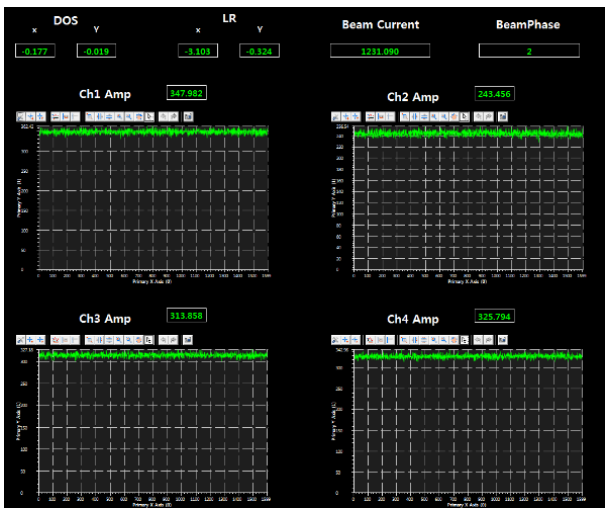


Figure 8: User interface of BPM using CSS.

EPICS Driver Support

The VME baseboard has a built-in EPICS IOC server and generates PV data, and serves it. The ADC data is supplied to the FPGA. The FPGA stacks the data in the FIFO memory in an IQ arrangement and hands over the data to the CPU board after one pulse. The EPICS driver support is implemented to extract the beam position, phase, and current. The beam position is different of sum and log ratio. Position mapping data of BPM is measured to obtain a calibration value, and application to the DAQ system is completed. The beam phase is vector sum of four signals from four electrodes in a beam chamber. The beam current is amplitude sum of four signals from four electrodes. Primary data processing is performed in the driver support area of the EPICS layer. Amplitude and phase are generated from IQ data. The FPGA code must be general and flexible, so it can be used for all BPM types with little modification.

The final value calculated and obtained from the driver support is registered in the EPICS PV. BPM data is monitored in real time through the Control System Studio (CSS) and Channel Access (CA) in shown Fig. 8.

CONCLUSION

The BPM DAQ system for KOMAC Linac and beamline has been developed. The MVME3100 and PENTEK7142 ADC board have been adopted to directly collect signals from the BPM 4 channels and process data. EPICS IOC for MVME3100 has been developed with the PENTEK driver. The BPM data processing is performed by EPICS driver support, and beam position, phase and current values could be checked in real time for each beam pulse. Libera spark HL products have been adopted and installed for beamline BPM. Libera is judged to be a user-friendly system in beam position electronics based on system on chip platform.

The VME baseboard has been adopted and used as a major device for KOMAC's high-performance electronics, including BPM, timing and LLRF systems. By unifying the platforms of various control system, short development processes and simple maintenance become the biggest advantages.

ACKNOWLEDGMENT

This work has been supported through KOMAC (Korea Multi-purpose Accelerator Complex) operation fund of KAERI by MSIT (Ministry of Science and ICT).

REFERENCES

[1] Yong-Sub Cho, Hyeok-Jung Kwon, Dae-Il Kim, Han-Sung Kim, Jin-Yeong Ryu, Bum-Sik Park, Kyung-Tae Seol, Young-Gi Song, Sang-Pil Yun, and Ji-Ho Jang, "The Komac Accelerator Facility," Proceeding of International Particle Accelerator Conference, Shanghai, China, May 2013, paper WEOBB101, pp. 2052-2054.

[2] Young-Gi Song, J. Korean Phys. Soc. 66, 449 (2015).

[3] J. Pietarinen, "VME Event Receiver (VME-EVR-230RF), Event Generator, Technical Reference", 2011.

[4] Experimental Physics and Industrial Control System (EPICS). http://www.aps.anl.gov/epics

[5] Libera Spark HL, http://i-tech.si

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV028

THE CONTROL AND ARCHIVING SYSTEM FOR THE GAMMA BEAM PROFILE STATION AT ELI-NP*

G. Chen †, V. Iancu, C. Matei, F. Ramirez, G. Turturica
Extreme Light Infrastructure – Nuclear Physics, IFIN-HH, Magurele-Bucharest, Romania

Abstract

The Variable Energy Gamma (VEGA) System of Extreme Light Infrastructure - Nuclear Physics (ELI-NP) is based on the Inverse Compton Scattering of laser light on relativistic electron bunches provided by a warm radio-frequency accelerator. The system will deliver quasi-monochromatic gamma-ray beams with a high spectral density and a high degree of linear polarization. The Beam Profile Station, which will be used for finer target alignment and spatial characterization of the gamma-ray beam, is one of the diagnostics stations under implementation at ELI-NP.

An EPICS Control and Archiving System (CAS) has been developed for the Beam Profile Station at ELI-NP. This paper describes the design and the implementation of the EPICS CAS for the Beam Profile Station, including the device modular integration of the low-level IOCs for the CCD camera Trius-SX674 and Mclennan PM600 Stepper Motor Controller, the design of the high-level GUI for real-time image acquisition and motion control, as well as the configuration of the archiving system for browsing the historic images and parameters.

INTRODUCTION

The Variable Energy Gamma (VEGA) System of Extreme Light Infrastructure - Nuclear Physics (ELI-NP) will produce intense gamma-ray beams with a spectral density higher than 0.5 x 10^4 photons/eV/s, a relative energy bandwidth better than 0.5%, high degree of linear polarization at more than 95%, and energy continuously variable from 1 MeV up to 19.5 MeV based on the laser Compton backscattering of laser photons off a relativistic electron beam.

The Gamma Diagnostics Stations are designed and under implementation to measure and to monitor the gamma beam diagnostics features [1]. To optimize the operation of the ELI-NP VEGA and its use for experiments, it is necessary to have the proper means to accurately predict the spatial, spectral and temporal characteristics of the gamma beam. The ELI-NP Gamma Diagnostics Stations are dealing with the equipment and techniques meant to optimize the gamma beam in order to make it available for user experiments within required parameters.

As a part of the Gamma Diagnostics Stations, the Gamma Beam Profile Station utilizes a CCD camera to collect the light produced in a scintillator placed in the beam by means of a mirror. The system is designed to be mobile and could be moved to different experimental areas.

The camera will be placed outside vacuum in a special mount, which ensures position reproducibility and easy removal. The scintillator and the mirror will be housed in a light-tight box inside the beam transport pipe. To ensure position reproducibility of the scintillator, a motor controller will be attached to the scintillator.

The CCD camera Trius-SX674 and the Stepper Motor Controller Mclennan PM600 have been chosen to be integrated into the CS of the Gamma Beam Profile Station.

Besides, the archiving system is required to store the PV samples periodically into an RDB.

SYSTEM DESIGN AND CONFIGURATION

The EPICS [2] based CAS for all the diagnostics stations will be designed and implemented by ELI-NP, to provide the machine information connection with the VEGA CS, to collect data from devices in the diagnostics stations, to monitor status of the devices, and to provide the High-Level Software (HLS) for the Gamma Beam Diagnostics.

General Architecture Model

The architecture of the CAS is structured as the standard three-tier structure:

User interface. These are graphical and non-graphical user interfaces. Most or all of these will be in the control room.

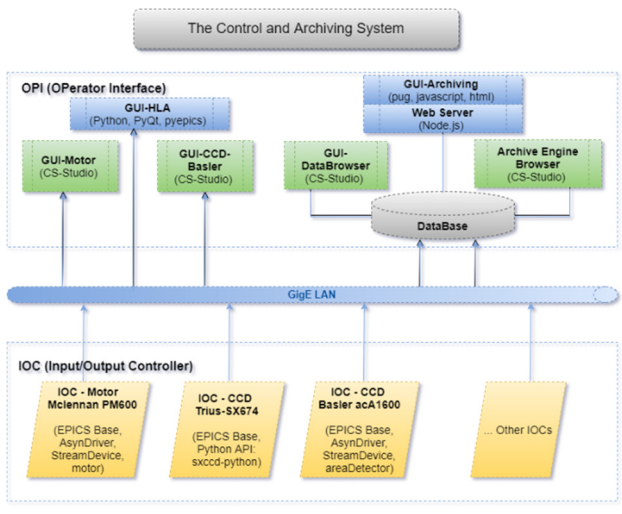


Figure 1: The software architecture of the Control and Archiving System.

Central services. Central services need to be run continuously irrespective of user activities, e.g., for archiving process variables' values, monitoring alarm states, etc.

Equipment interfaces. This tier is responsible for interaction with equipment and devices. It provides an abstract

* The work is supported by ELI-NP Project (http://www.eli-np.ro/)
† guangling.chen@eli-np.ro

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV028

representation of equipment to higher layers through which the equipment can be monitored and controlled. The elementary entity on this level is called an Input Output Controller (IOC).

The gigabit Ethernet (GigE) Local Area Network (LAN) allows the communication between the IOCs, Archivers, and OPIs.

The software architecture of the control and archiving system is shown as in Fig. 1. The EPICS IOC software package are dealing with the low-level input and output controls to the devices, while the GUI (Graphical User Interface) for HLA (High-Level Application) and the GUI for Archiving system are taking care of the High-level user operations.

Design of the Control System

The control of the Beam Profile Station for Gamma Beam diagnostics includes one Mclennan PM600 motor controller and two Trius-SX674 CCD cameras.

IOC Design for the Motor Controller The Mclennan PM600 is a powerful single axis motion controller which has been selected for the beam profile station to move the scintillators in and out of the gamma beam.

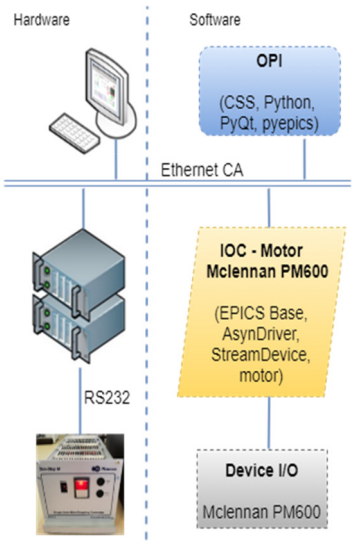


Figure 2: The hardware and software structure for the motion control.

The motor controller Mclennan PM600 is connected to the PC via a RS232 serial cable, and to the linear bellows drive via the dedicated Motor and Limit/Datum cables.

The diagram showing above (Fig. 2) is the hardware and software structure for the motion control which includes a Mclennan PM600 controller, connected via RS232 serial cable to an IOC running EPICS and motor related software. The IOC is connected to the operator's workstations via the Ethernet network.

Mclennan PM600 controller is supported by the EPICS device support module motor record. By using the motor record, the IOC can read motor position from the controller's readback register, can get readback information while the motor is moving, and can trigger a forward link when a complete motion is finished.

IOC Design for the CCD Cameras The image acquisition of the Beam Profile Station for Gamma Beam diagnostics includes two Trius-SX674 CCD cameras which are connected to the PC via USB cables.

The goal is to provide a functional EPICS application using the required Python device support and device drivers for the Trius-SX674 CCD camera.

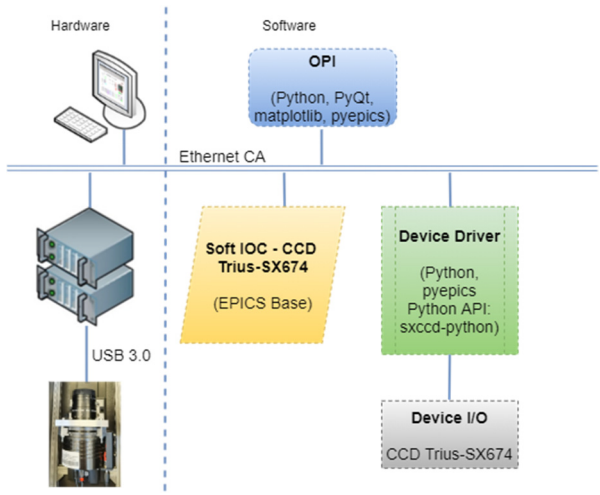


Figure 3: The hardware and software structure for the CCD camera.

The Hardware and Software structure design for the Camera (Fig. 3) includes a Trius-SX674 CCD camera, connected via USB 3.0 to an IOC running EPICS and camera related software. The IOC is connected to the operator's workstations via the Ethernet network.

Trius-SX674 CCD camera is not a standard genicam camera, so the most common used EPICS device support modules aravisGigE and areaDetector cannot be used as the Linux camera driver. We use Python API sxccd-python as device support module, and implement a driver wrapper based on this device support module. Besides, a soft IOC will deal with the EPICS databases for the CCD camera control.

Design of the OPI (GUI) Typically, areaDetector plugins are used to save images and to provide arrays used for displaying images acquired by cameras using the GUI. Since the Trius-SX674 CCD camera is not supported by EPICS areaDetector, the GUI need to be implemented individually.

The EPICS device support module motor record provides a CSS BOY-based GUI to perform the device-layer control and troubleshooting for the Mclennan PM600 controller. Moreover, a Python and PyQt based GUI for the motion control and the image acquisition for the Beam Profile Station will be implemented.

The goal is to provide a functional EPICS based GUI to control the Mclennan PM600 controller and to acquire image from the Trius-SX674 CCD camera.

Design of the Archiving System

The architecture of the archiving system (as shown in Fig. 4) consists of an archive engine, the MySQL RDB, a web server and the web-based GUI.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV028

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The Archive Engine samples Process Variable (PV) data, time stamp etc., from EPICS IOCs via Channel Access (CA), and places them in a Relational Database. Users can then access the historic data from the database as well as the live data from the PV channels using the CSS Data Browser. Config.xml and settings.ini are the two basic configuration files needed to configure the database by using the provided ArchiveConfigureTool. Besides, PhpMyAdmin is used as the tool for the database maintenance and configuration.

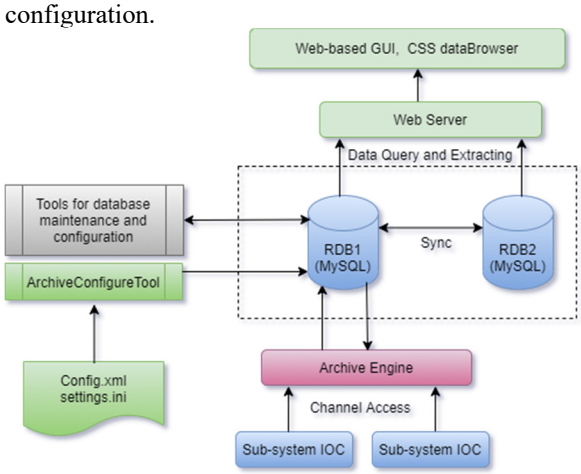


Figure 4: The structure and data flow of the archiving system.

IMPLEMENTATION

Implementation of the IOCs

To integrate the Trius-SX674 CCD cameras, we use Python API sxccd-python [3] as the device support module. Based on the sxccd-python module, a driver wrapper has been implemented for the communication with the soft IOC which deals with the EPICS databases for the CCD camera control.

The IOC for the CCD camera starts with an initialize module to deal with the USB connection and soft IOC initialization. The read-out loop detects when the image acquisition toggle is on, then the image will be captured and the waveform data will be put into the PV of image.

Mclennan PM600 controller is supported by the EPICS device support module motor record [4]. To integrate the PM600 controller, we use the version of the motor record motorR7-1, as well as other related modules in EPICS SynApps5-8.

The IOC for the motor controller is implemented following the standard steps for an EPICS motor IOC. The IOC start file st.cmd is configured to clarify the support module, to establish the serial connection, and to generate the database by the file motor.substitutions and motorUtil.db

Implementation of the OPIs

The operator interface, the GUI is implemented using python3 and PyQt5. Some other python modules, such as numpy, pyepics, threading, and matplotlib etc., are also imported for the development.

A screenshot of the GUI has been shown in Fig. 5.

The left part of the GUI is dealing with the following motion controls:

- Turn on/turn off the motor remotely.
- Move the motor for a certain distance.
- Set the velocity.
- Update the feedback in real-time.
- Calibrate the motor to the home position.

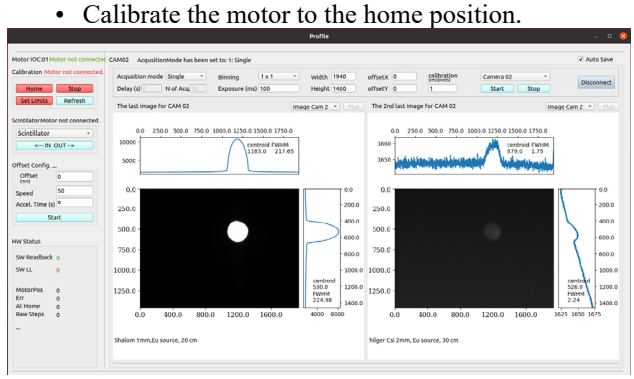


Figure 5: The GUI for the control of the motor and CCD.

The rest part of the GUI is able to deal with the image capturing controls as following:

- Turn on/turn off the camera remotely, to acquire image remotely from the client user interface.
- Set the width, height, exposure time, offset, and acquisition mode (single/continuous).
- Monitoring the status of the camera, to set the exposure time for the camera.
- Provide real-time processing of the images, e. g., profile for X and Y axis, centroid, and FWHM.
- Save images in different formats for the image analysis.
- Save the waveform data of the images.
- Save the parameters for the image capturing.

Implementation of the Archiving System

The EPICS Archive Engine Service in CS-Studio has been chosen as the software tool for the archiving system development. To configure the archive engine, Phoebus [5], the latest version of CS-Studio has been chosen as the archive engine configuration tool to read samples from PVs and write them to the RDB. The data browser has been chosen also to browse the historical or monitoring data from the RDB.

The Archive Engine has been configured using the ArchiveConfigTool. PhpMyAdmin has been used as the database management tool.

Besides, we stored on the database server all the images which are acquired from the Beam Profile System, as well as the image data, and the image related parameter files in the file system.

Implementation of the Web Server and Web-Based GUI

Figure 6 shows the scheme used to implement the web-based system using the MVC (model-view-controller) pattern. With this implementation, the system can serve requests in two ways: by using the embedded views and also via API's fetching and sending raw data.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV028

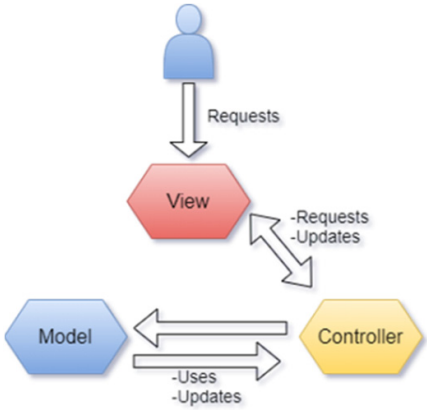


Figure 6: MVC representation.

The web server and the web-based GUI for the archiving system is implemented based on Node.js, Express, MySQL, Sequelize, Javascript, Pug, HTML, and VS code.

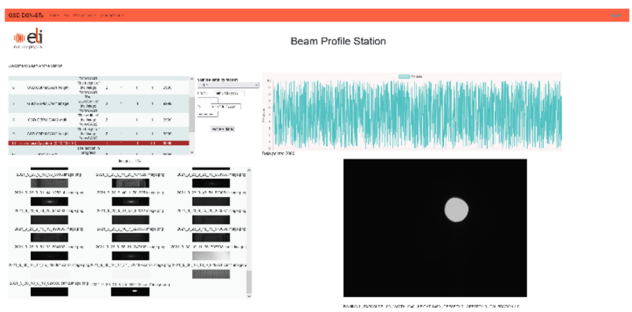


Figure 7: EPICS-based archive application GUI.

Figure 7 shows the results of HTTP requests using the embedded view (GUI). By using the web-based GUI, users can connect to the archive RDB which stores the data fetched by the archive engine. Users can also browse the data by choosing a PV from the PVs-list and then plot the data immediately. Moreover, the GUI allows users to choose an image from the images-list and to display it immediately with the parameters associated to the image.

CONCLUSION

The EPICS based control and archiving system has been implemented for the Beam Profile Station at ELI-NP. The devices have been integrated into the control system, and the archive engine has been configured for the archiving system.

The IOC for the CCD camera has been configured based on sxccd-python. A driver wrapper has been implemented for the communication with the soft IOC of the camera. The IOC for Mclennan PM600 controller has been configured using the version of the motor record motorR7-1, as well as other related modules in SynApps5-8. The GUI for the control has been implemented using python3 and PyQt5.

The archive engine has been configured using CSS Phoebus. The web server and the web-based GUI have been implemented using the MVC pattern.

This implementation of the control and archiving system for Beam Profile Station has been proved as a professional programming structure that helps in maintenance and readability of the whole system, moreover, it is reliable enough for scaling up in the future implementation for other diagnostics stations such as Beam Energy Station and Beam Polarization Station at ELI-NP.

ACKNOWLEDGEMENT

The work was supported by the Project Extreme Light Infrastructure - Nuclear Physics (ELI-NP). The author would like to thank Dr. Marian TOMA for his valuable comments and suggestions.

REFERENCES

[1] H.R. Weller *et al.*, "Gamma Beam Delivery and Diagnostics", *Romanian Reports in Physics*, vol. 68, pp. S447–S481, 2016. http://www.rrp.infim.ro/IP/S9.pdf

[2] EPICS, http://www.aps.anl.gov/epics/

[3] Python API for Startlight Xpress CCD (sxccd), https://github.com/leeneil/sxccd-python

[4] EPICS: Motor Record and Device/Driver support, https://epics.anl.gov/bcda/synApps/motor/

[5] CS-Studio Phoebus, https://controlssoftware.sns.ornl.gov/css_phoebus/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV030

REDESIGN OF THE VELO THERMAL CONTROL SYSTEM FOR FUTURE DETECTOR DEVELOPMENT

S. A. Lunt*, University of Cape Town, Cape Town, South Africa
L. Zwalinski, B. Verlaat, CERN, Geneva, Swizerland

Abstract

The Detector Technologies group at CERN has developed a Two-Phase Accumulator Controlled Loop (2PACL) [1] test system for future detector development, using reused hardware from the LHCb Vertex Locator (VELO) Thermal Control System [2]. The fluid, electrical and control systems have been redesigned and simplified by removing redundant components because it is no longer a critical system. The fluid cycle was updated to allow both 2PACL and integrated 2PACL [3] cycles to be run and the chiller was replaced with an air-cooled unit using hot gas bypass to achieve a high turndown ratio. The electrical systems were upgraded to improve usability and practicality. The control system logic is being developed with the CERN's Unified Industrial Control System (UNICOS) framework. This paper presents the details of the design and implementation.

INTRODUCTION

The Vertex Locator (VELO) Thermal Control System (VTCS) is the first CO_2 detector cooling system used at CERN to cool LHCb's VELO sub-detector. Following the successful use of CO_2 cooling in the VTCS and AMS-02 [4] on the International Space Station the number and capacity of CO_2 based detector cooling systems has increased significantly with installations in the LHCb, ATLAS and CMS experiments at CERN. The VTCS was retired and replaced with a new system in 2019 [5]. The Detector Technologies (DT) group decided that the system could be refurbished and used for the development and testing of future detectors for ATLAS on the surface.

The system was partially redesigned to prepare it for the new role. The electrical system was replaced to ensure it met the department's standards. The control logic has been rewritten with a new program developed under the Unified Industrial Control System (UNICOS) framework [6]. The fluid systems have remained largely unaltered with only the position of the accumulators being changed. All redundant components have been removed and a mode that runs the integrated two-phase accumulator controlled loop (I2PACL) [7] have been added. The chiller has been replaced to make it more suitable for surface applications without available underground infrastructure.

FLUIDIC SYSTEM

The original fluid systems were designed with redundancy in each layer to make sure no single failure would cause the detector's cooling supply to stop because the silicon tracking sensors of the detector degrade quickly at temperatures above 0 °C after they have been exposed to radiation. These redundant elements have been removed from the new system because a failure of the cooling system in a test environment is acceptable because the sensors have not been exposed to radiation [8].

The VELO detector consists of two halves that sit on either side of the beam. The VTCS was designed with two independent, two-phase accumulator controlled loops (2PACL), one for each side. A third pump was installed to act as a backup for either side in the event of a pump failure [2]. This pump along with the pipes and valves that supplied it have been removed. Two valves were left in because they provide mechanical support for the surrounding pipes and convenient access for maintenance of the system.

The new system will retain the two independent 2PACL, shown in Fig. 1, loops to maximise its versatility and minimise the changes that need to be made to the CO_2 system. To improve the responsiveness of the controllers to changes in the process, in-flow temperature sensors are being added. The accumulator heat exchanger previously connected to the backup chiller will be used to allow the system to run in the simpler Integrated 2PACL (I2PACL) mode [7].

Integrated 2PACL

Since the development of the VTCS the DT group has made improvements to the two-phase process: integrated 2PACL. This removes the need to actively cool the accumulator, reducing the complexity of the fluid and control systems. In an I2PACL system, the sub-cooled CO_2 leaving the pump flows through a heat exchanger in the accumulator, shown in Fig. 2.

Removing the need for a controlled external cooling source simplifies the control of the accumulator's saturation pressure, thus allowing the saturation temperature of the accumulator to be controlled with the heater alone. The heater provides the energy equal to the heat capacity of the liquid CO_2 and by controlling the power the pressure is regulated (Fig. 3).

The accumulators contain two condensing spirals, which were used by the main and backup chillers. One of these condensing spirals will now be used for the I2PACL mode; the other will be supplied from the chiller to control pressure in 2PACL mode.

Chiller

The primary water-cooled chiller and backup air-cooled chiller, will not be reused. Cold water is a service supplied in the caverns at CERN but is not readily available in other

* lntstu001@myuct.ac.za; slunt@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV030

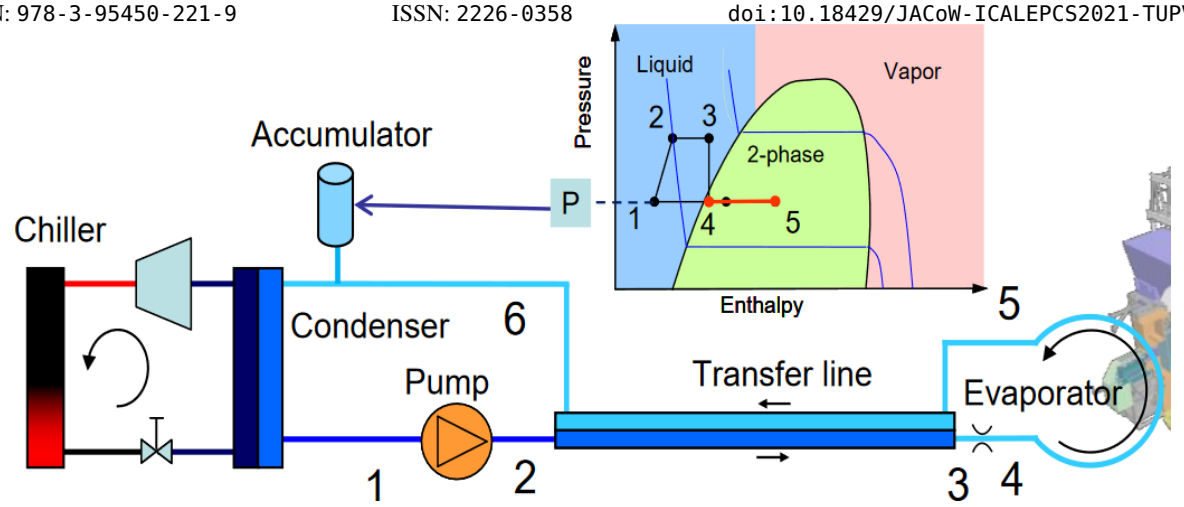


Figure 1: 2PACL system diagram [9].

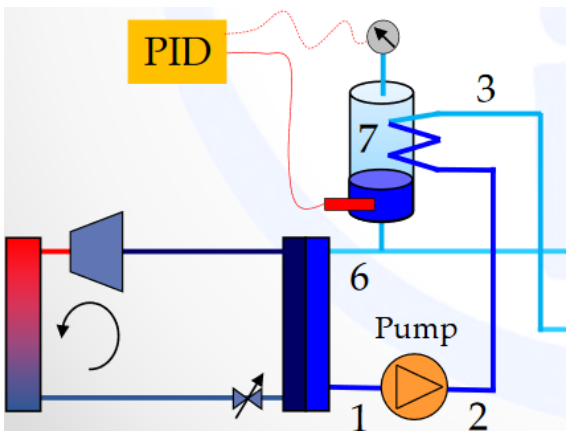


Figure 2: I2PACL system [10].

parts of the site, making the old primary chiller problematic to re-use on the surface.The backup chiller's capacity is only 1 kW which is too low. They will be replaced with a new 3.15 kW air-cooled chiller.

The chiller is based on a commonly used design that the group first developed for the Light Use Cooling Apparatus for Surface Zones (LUCASZ) systems, shown in Fig. 4 [10]. Hot gas bypass and cold liquid injection valves allow the chiller to continue running when varying or low loads are applied to it. This is unusual in refrigeration systems as they are typically designed to operate at a fixed load. In this case, the load will be variable and the chiller needs to be capable of running normally at any load that is applied to the system.

The chiller will use R449a to replace R404a which is prohibited from being used in new systems from 2020 by the European Fluorinated Gas Regulation to reduce the climate impact of refrigeration systems [11]. The adaptation to R449a required some minor modifications to the design and control to accommodate the differences in the characteristics of the two refrigerants.

ELECTRICAL SYSTEM

The electrical systems were examined to determine what could be re-used from the original system and what would require upgrading. The electrical hardware in VELO was installed in rack mounted boxes with connectors on the back. These boxes were replaced with a simplified electrical system where all the electrical components are mounted vertically in the cabinet behind the front door, as shown in Fig. 5.

This design is advantageous because it makes the components more visible and accessible. The status of circuit breakers can be seen and set or reset without disconnecting any cables or removing components from the cabinet. The flat layout also allows for signal cables and power cables to be more easily isolated from each other, reducing the impact of electrical noise on other components. A new electrical design was developed using the original system as a baseline. The design was constrained by the available area of the panel, 500×1500 mm, and by the project budget. Using components that are in good condition from the VTCS or were in spare stock at the time will reduce the cost.

CONTROL

The control architecture has been completely redesigned using CERN's UNICOS continuous process control framework [6] to make sure it meets CERN's standards and will be easily maintainable and modifiable in the future. The system will be controlled through Siemens WinCC OA for supervisory control and data acquisition (SCADA) which will operate on CERN's Technical Network (TN) as shown in Fig. 6.

The PLC is a Siemens S7-400 station that will be reused with changes to some modules. The CPU was upgraded to one with a greater work memory to accommodate the larger size of programs developed using UNICOS. An incompatible controller card was removed and replaced with an analogue output (AO) module. The current system configuration is shown in Table 1.

The spare channels were retained because we already had all the IO modules apart from the analog outputs and

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV030

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

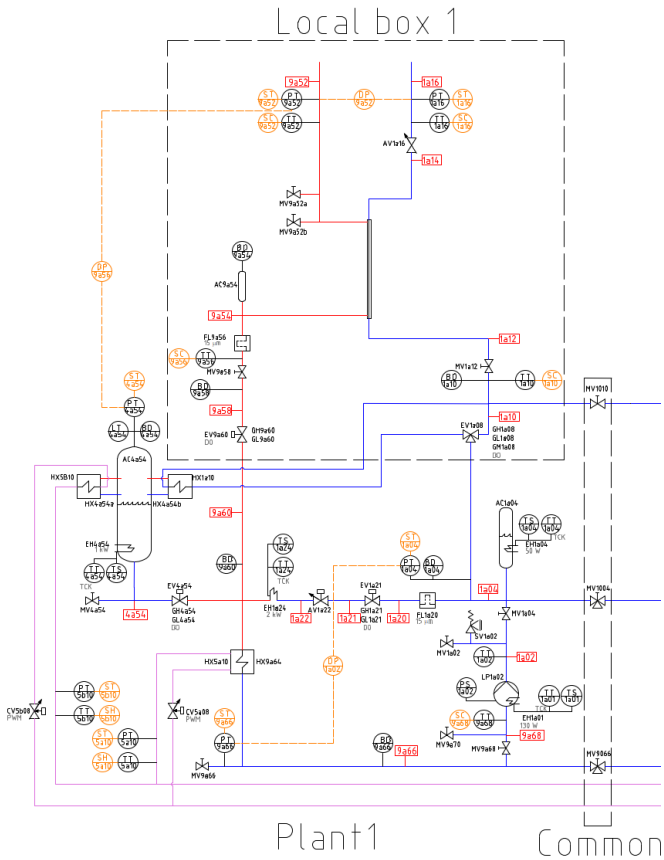


Figure 3: CO_2 P&ID diagram.

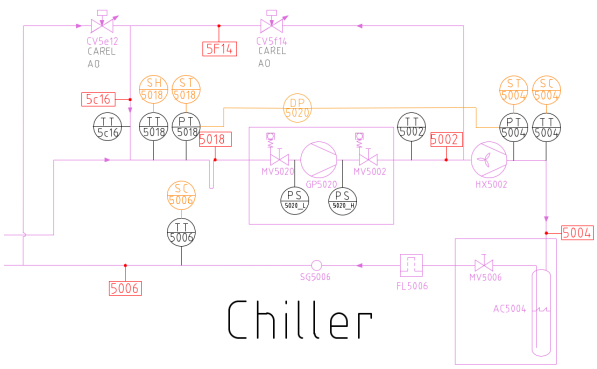


Figure 4: Chiller P&ID.

Table 1: PLC IO Channels

IO type	Used	Total
Digital input	63	96
Digital output	41	96
Analog input	47	64
Analog output	2	8

they were connected to IO terminals in the cabinet. They also provide a comfortable buffer against the failure of an individual card. The S7-400 platform is reaching end of life, if a IO card failed it would be possible to switch to a spare unit and continue operating while a replacement is found.

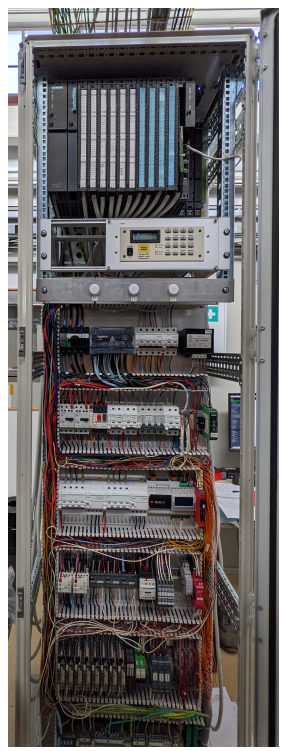


Figure 5: New power distribution panel.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV030

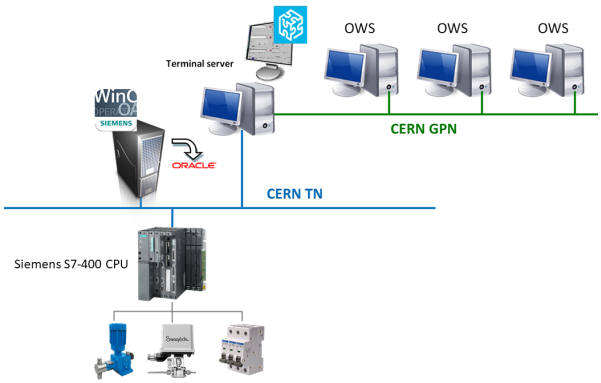


Figure 6: Control system architecture.

Program Structure

UNICOS applications are made up of process control objects (PCO) each representing a separate unit within the system. A PCO can be made up of actuators or other PCOs as shown in Fig. 7. This system has a top-level PCO called CO2. The chiller is its own PCO containing all its actuators. Each 2PACL system is identical and has it's own PCO which contains the CO_2 actuators and allows the systems to operate independently. Each local box has its own PCO which is contained by the respective plant PCO and controls the flow of CO_2 from that PCO.

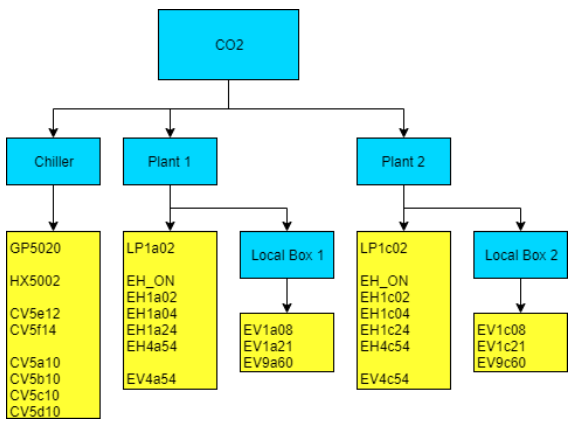


Figure 7: PCO structure.

The state of a PCO will be determined by the state of the parent PCO. If this is in Stop mode then the dependant PCO's will also be stopped. If a parent PCO is in Run mode, the dependant PCO's will be allowed to run subject to its normal run conditions. The two CO2 plants can run independently of each other. The chiller will start when one or both CO2 plants are requested to run.

Safety

The use of liquid CO_2 in a closed system introduces several important safety concerns which the system design must mitigate. The control system must ensure that no cold liquid CO_2 is trapped in a confined space. If this happens the pressure will increase dramatically as the CO_2 heats and boils, causing the pipe to burst or safety systems to activate and the CO_2 will be vented from that section. This could cause damage to the system and harm anyone who is in the vicinity.

Experiments will be manually connected to and disconnected from the local box. This process requires manual intervention by the operator. The design must protect both the operator and the hardware from potential operator error. If the operator opens a local box valve at the improper time CO_2 could be vented into the atmosphere. This poses an asphyxiation hazard to the operator.

The Stop or Safe position of controlled valves is set to open except for the local box shutoff valves, which will close. This ensures that there is always a path for CO_2 to return to the accumulator, but isolates the local box from the rest of the system. When an experiment is disconnected, the PLC will wait until only vapour is present in the transfer line before allowing the operator to continue.

If the system is stopped without disconnecting the experiment or an emergency stop is triggered, the local box shut-off valves will close with liquid still inside. This is important to simplify the shutdown procedure and allow an experiment to remain connected. A pressure vessel, AC9a54 shown in Fig. 3, will be added to allow any liquid CO_2 trapped in the local box to pressurise without causing any damage. It will have the same volume as the tubing in the local box and be placed above the accumulator and transfer line. This ensures that there is only vapour present at all times.

Start-Up and Experiment Connection

The chiller and both CO_2 plants have no start-up sequences. They start running when that state is requested and any interlocks are cleared. The Local Box has a more detailed process depending on the mode it is running in. These are shown in Fig. 8 with the transitions explained in Table 2.

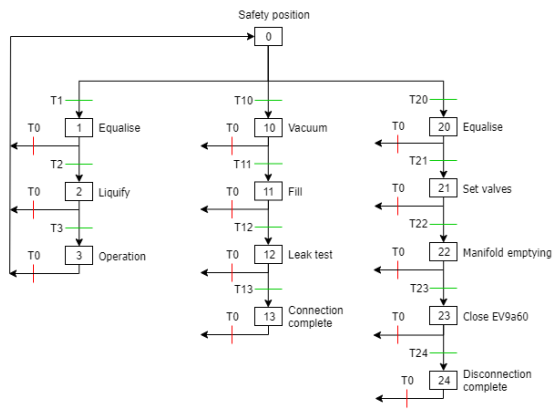


Figure 8: Stepper for the Local Box PCO.

Normal operation For normal operation of the local box the local box PCO changes the set point of the CO_2 plant to equalise the pressure difference across the valves before opening. Following this, the loop is liquefied before the set point is changed back to the user set point.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV030

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 2: Transition Condition Descriptions

Transition	Condition description
T0	Stop request
T1	Device connected and run mode
T2	Pressure difference within limits
T3	Liquid CO2 is circulating
T10	Device disconnected and connect mode
T11	Vacuum applied
T12	Device filled with CO2
T13	Leak test OK
T20	Device connected and disconnect mode
T21	Pressure difference is within limits
T22	CO2 return valve open
T23	Super-heating in the local box
T24	Plant valves closed

Connection procedure The connection process is a manual process that is performed by the user. The PLC uses the sensors in the Local Box to monitor the steps that the user performs and ensures that they are all followed properly.

Disconnection procedure To disconnect an experiment, the Local Box is first emptied of liquid CO_2 . This is done by reducing the accumulator set point to its minimum allowable value and waiting for super-heating to be detected in the local box. This removes as much CO_2 from the local box as possible and allows the user to vent and disconnect the local box safely.

CONCLUSION

The VTCS has been significantly simplified and optimised to fulfil its new role for the group. The new design follows the most recent norms and standards of CERN and the DT group to ensure that it can be maintained or modified as needed to continue supporting the group's research.

ACKNOWLEDGEMENTS

I would like to thank Lukasz Zwalinski, Bart Verlaat, Michal Zimny and Wojciech Hulek for their continuous guidance on this project. Eric Thabuis and Loic Curtat from the gas workshop for their assistance with technical elements of the rebuilding of the system. Pawel Bak contributed to the development of the electrical schematics.

APPENDIX

The system documentation will be uploaded to the Engineering Data Management System (EDMS) and will be available at: https://edms.cern.ch/. The project is named DTCS and can be found under: CERN-0000219585.

REFERENCES

[1] B. Verlaat, M. V. Beuzekom, and A. V. Lysebetten, "CO_2 cooling for HEP experiments," p. 9, 2008. https://cds.cern.ch/record/1158652/files/p328.pdf

[2] B. Verlaat, "Technical Description of the Velo Thermal Control System," 2010. https://edms.cern.ch/ui/file/1056946/3.0/VTCSdesignReport31aug10_EDMS1056946ver3p0.pdf

[3] B. Verlaat, "Compact cooling system and method for accurate temperature control," European pat. 2753887B1, 2020. https://patents.google.com/patent/EP2753887B1/en?oq=EP2753887

[4] A. A. M. Delil, A. A. Woering, and B. Verlaat, "Development of a Mechanically Pumped Two-Phase CO_2 Cooling Loop for the AMS-2 Tracker Experiment," in *International Conference On Environmental Systems*, 2002, paper 2002–01–2465. doi:10.4271/2002-01-2465

[5] H. Schindler, "LHCb upgrade installation workshop in view of the long shut-down 2," 2018. https://indico.cern.ch/event/687068/#14-cooling-plants-and-dry-gas

[6] H. Milcent, E. Blanco, F. Bernard, and P. Gayet, "Unicos: An open framework," in *Proc. ICALEPCS'09*, Kobe, Japan, 2009, pp. 910–912. https://jacow.org/icalepcs2009/papers/thd003.pdf

[7] B. Verlaat, "Compact cooling system and method for accurate temperature control," U.S. Patent 20140230471A1, 2011. https://patents.google.com/patent/US20140230471A1/en

[8] P. Jałocha, "LHCb VELO Detector Cooling Requirements," 2016. https://indico.cern.ch/event/487201/contributions/1997116/subcontributions/180176/attachments/1227161/1797159/LHCb_VELO_Detector_Cooling_Requirements.pdf

[9] B. Verlaat and A. P. Colijn, "CO_2 Cooling Developments for HEP Detectors," in *Proceedings of VERTEX 2009 (18th workshop) — PoS(VERTEX 2009)*, vol. 95, 2010, p. 031. doi:10.22323/1.095.0031

[10] B. Verlaat, "LUCASZ user training for FPIX," 2017.https://indico.cern.ch/event/688678/

[11] *Regulation (EU) no 517/2014 of the european parliament and of the council of 16 april 2014 on fluorinated greenhouse gases and repealing regulation (EC) no 842/2006 text with EEA relevance*, Legislative Body: CONSIL, EP, 20, 2014. http://data.europa.eu/eli/reg/2014/517/oj/eng

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV031

LHC VACUUM SUPERVISORY APPLICATION FOR RUN 3

Sebastien Blanchard†, Ivo Ambrósio Amador, Nikolaos Chatzigeorgiou, Rodrigo Ferreira, Joao Diogo Francisco Rebelo, Paulo Gomes, Christopher Viana Lima, Gregory Pigny, Andre Paiva Rocha, Lampros Zygaropoulos
CERN, Geneva, Switzerland

Abstract

The LHC Vacuum Supervisory Control and Data Acquisition application has been upgraded to fulfil the new requirements of Long Shutdown 2 and Run 3.

The number of datapoint elements has been increased from 700k to 1.5M, which constitutes a challenge in terms of scalability. The new configuration of pumping station control hardware has led to an increase in the number of permanently connected PLCs – from 150 to almost 300. A new concept has been developed and deployed, in which the PLC configuration is updated online. The goals were to automate, and to speed up periodic updates of the control system. Integrating of the wireless mobile equipment had led to the acquisition of expertise in dealing with temporary connections and dynamic insertion of device representation in the synoptic.

Other new features include: the introduction of an innovative remote control and representation in synoptic panel of hardware interlocks, the development of a pre-configured notification system, and the integration of asset management into the user interface.

VACUUM SYSTEMS

The Large Hadron Collider (LHC) has two types of vacuum systems:

- The beam vacuum system, which reaches ultra-high vacuum (below 1×10^{-9} mbar).
- The insulation vacuum system, used for thermal insulation of cryogenic components, and which reaches high vacuum (below 1×10^{-5} mbar).

Both systems are divided into sectors delimited by vacuum valves or windows. Vacuum sectors will reduce or entirely prevent the propagation of sudden pressure increases. They also allow for independent venting and interventions.

The beam vacuum system of the LHC has 321 sectors. Only a few sectors have pumping group stations [1] permanently installed. For most sectors, pumping group stations – and sometimes, bake-out cabinets [2] – are temporarily installed during installation or interventions. Pumping group stations achieve high vacuum, after which bake-out cycles decrease the outgassing rate of the vacuum vessel and activate the NEG thin-film coatings. Finally, the permanently installed pumps are started. These include, for example – ion pumps. In addition, every sector has a full set of vacuum gauges.

The insulation vacuum system of the LHC has 235 sectors. These sectors have pumping group stations [1] and vacuum gauges permanently installed.

† sebastien.blanchard@cern.ch

Table 1: List of Remote-controlled Instruments

Type	System	Count
Sector Valves	Beam	324
Sector Valves	Insulation	68
Fixed Pumping Groups	Beam	10
Fixed Pumping Groups	Insulation	193
Ion pumps	Beam	906
Pirani Gauges	Beam	455
Pirani Gauges	Insulation	343
Penning Gauges	Beam	803
Penning Gauges	Insulation	343
Ion Gauges	Beam	196
Membrane Gauges	Insulation	237

Table 1 lists the individual quantities of remote-controlled vacuum instruments in the LHC.

HARDWARE ARCHITECTURE

The automation layer of the vacuum control hardware architecture is PLC-based. Figure 1 shows the hardware architecture.

PLCs and controllers are installed in radiation-free areas of the LHC's underground, while the radiation-tolerant measuring cards [3], for gauges, are installed in the tunnel – close to the instruments. Table 2 lists the individual quantities of PLCs, controllers, measuring cards and hardware interlocks installed in the LHC.

Table 2: List of Controllers

Type	System	Count
PLCs	Beam	40
PLCs	Insulation	253
PLCs for mobiles	Beam/Insul.	350
Commercial controllers	Beam/Insul.	654
Valve controllers	Beam	324
Radiation tolerant cards	Beam/Insul.	881
Hardware Interlocks	Beam	282
Hardware Interlocks for Cryogenics	Insulation	539

After a successful Run 2, the LHC entered into Long Shutdown 2 (LS2). This will have lasted from the end of 2018, to the beginning of 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

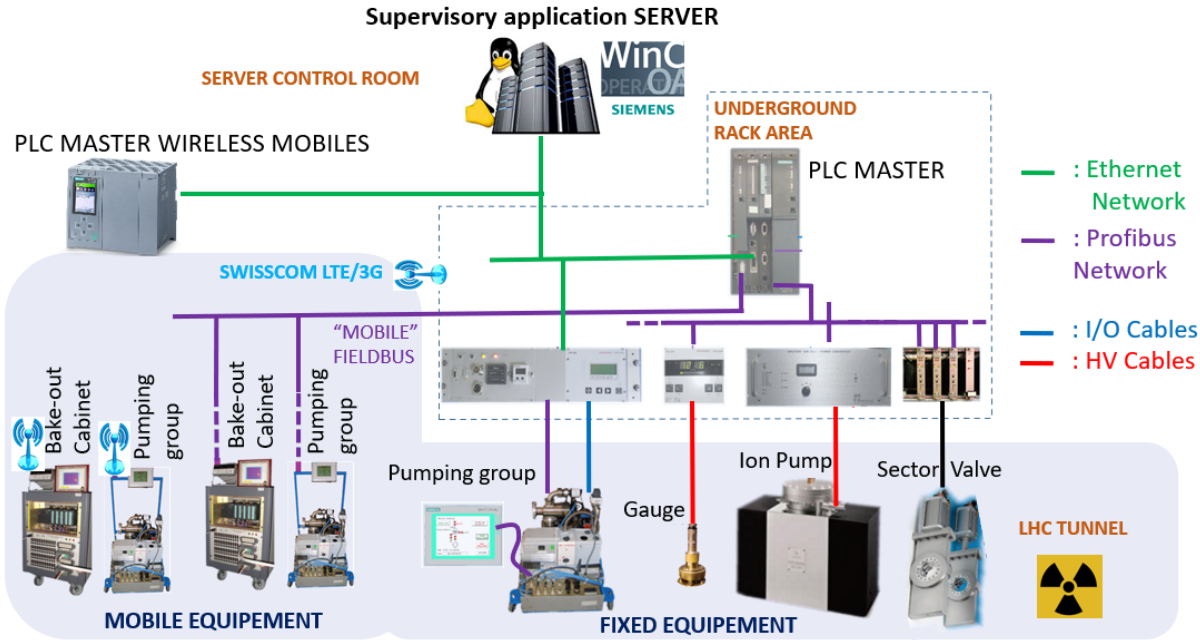


Figure 1: Control hardware architecture.

The following hardware has been upgraded during Long Shutdown 2:

- The permanently installed pumping group controllers (120 models of S7-300, to be replaced with 219 models of S7-1200)
- The insulation vacuum gauges and PLC interlocks (120 models of S7-300, to be replaced with 34 models of S7-1500)
- Mobile pumping groups and bake-out cabinets have been fitted with a 3G/LTE communication solution (55 units).
- New, radiation-tolerant measuring cards (881 units) have been installed for membrane, Pirani and Penning gauges.

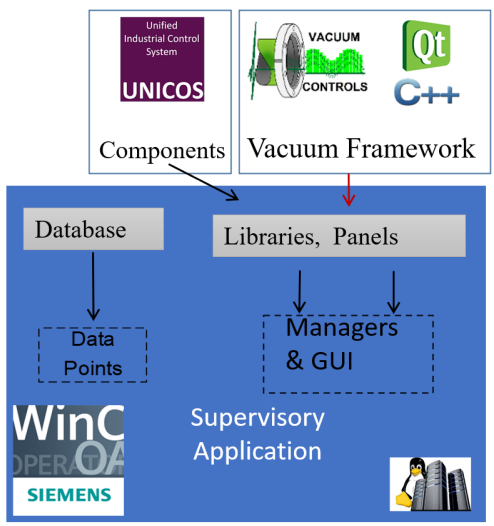


Figure 2: The supervisory application and its components.

THE SUPERVISORY APPLICATION

The Supervisory Control and Data Acquisition of the LHC vacuum system is based on a commercial product: WinCC OA™ by Siemens™. The Figure 2 shows the application components: managers, drivers and libraries. These come from either WinCC OA™ native components, or from CERN standard frameworks such as UNICOS [4], JCOP [5], and the Vacuum Framework [6].

The Vacuum Framework (libraries, panels, and components) customises the application to facilitate vacuum devices and features required for operating the LHC vacuum system.

Proof of Scalability

During the period of Long Shutdown 2, the vacuum supervisory application of the LHC had seen successful growth on an unprecedented scale. The number of data points has increased from 700k to 1.5M due to:

- Integration of 3G/LTE communication solution.
- Integration of new, highly parameterised controllers such as the Agilent™ Ion pump controller.
- Increase of device number, vacuum sectors number and layout complexity.
- Increase of PLC connections, from 150(before Long Shutdown 2) to 290.

DEVELOPMENT OF NEW DEVICE TYPES

Three major control types have been completely refactored (pumping group, ion pump and rough to high vacuum gauge).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV031

Pumping Group Device Type

A pumping group station is, at minimum, composed of a primary pump and a valve. Depending on the subtype, it can contain a turbo-molecular pump, four additional valves, one or two gauges, an integrated bakeout system, a heated exhaust system, etc. The process is controlled by a PLC, which also manages the interlocks.

The pumping group can be either permanently installed, or temporarily connected to the vacuum vessel (the so-called mobile pumping group). Mobile pumping groups are remote-controlled – either through a Profibus network deployed during the LHC installation in the 2000s, or through the new 3G/LTE communication solution [7].

Altogether, it represents a very large number of subtypes due to all the possible combinations. The development guideline was to standardise the behaviour of the device representation in the synoptic panel and the layout of the device details panel. For all the temporary (mobile) or permanently installed pumping group in the LHC accelerator, a single device type (set of libraries) has been developed.

Process specifications did not change since LHC Run 2. There were, however, significant changes to pumping group control. During LHC Run 2, every instrument (pumps, valves, gauges, etc.) had been controlled individually, and the pumping group had been managed globally, by a process. This allowed the switching of instruments to manual operation, and to drive them independently. As a result of the pumping group controller upgrade, the control becomes global-only, without the possibility of switching individual instruments to manual operation.

At first, development of the supervisory device libraries started in absence of new specifications. Indeed, the instrument itself had not changed; therefore, new specifications had not been requested. As mentioned above, however, the pumping group control method was very different. Choices regarding the new behaviour and standardisation had to prioritise speed over consensus, considering the strict deadline for first deployment.

The first deployment – even though it had allowed for perfect remote control of the pumping groups – was not a success. Mainly due the lack of status information in the synoptic panel, which confuse an operator who had been accustomed to the previous behaviour. A new version must be developed – this time with proper specifications, approved by the operators responsible. The next development and deployment fixed all the issues.

The 3G/LTE communication mobile pumping group is managed by the same device type as the permanently installed pumping group. A new dynamically built synoptic panel has been developed [8] and deployed. The synoptic library inserts or removes the widget representation of the mobile device changing dynamically the length and organisation of the synoptic panel according to the device connection state. The first deployment of the new, dynamically built synoptic panel experienced some issues. The rebuild and display of the synoptic takes a couple of seconds, which is acceptable in normal conditions. Unfortunately, however, the application had suffered continuous disconnection and re-connection cycles, resulting in the synoptic rebuilding all the time. This issue has been fixed by, first, invalidating the device (with a blue colour animation), then introducing a ten-minute timeout, before finally removing the device representation and rebuilding the synoptic.

New Strategy for Pumping Group Updates

Concerning the new hardware, each pumping group has its own PLC controller. This means 219 PLCs that may require updating during LHC Shutdowns and Technical Stops.

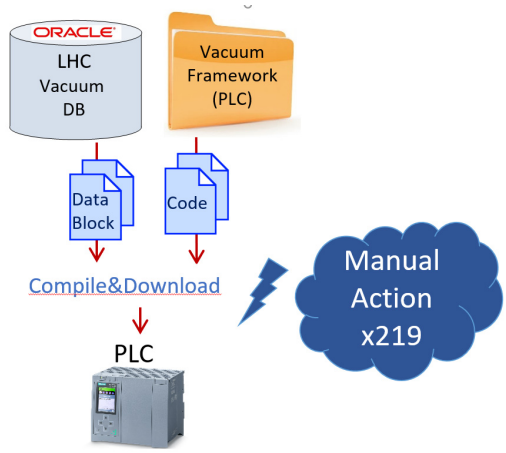


Figure 3: PLC update procedure, following the old/standard strategy.

The update describes in Figure 3, following the standard procedure (as was common before the new strategy) would have taken a very long time. 219 compilations and downloads - each performed manually - usually undertaken for the trivial purpose of changing equipment names or process parameters. Furthermore, this procedure also requires that pumping groups be vented and stopped.

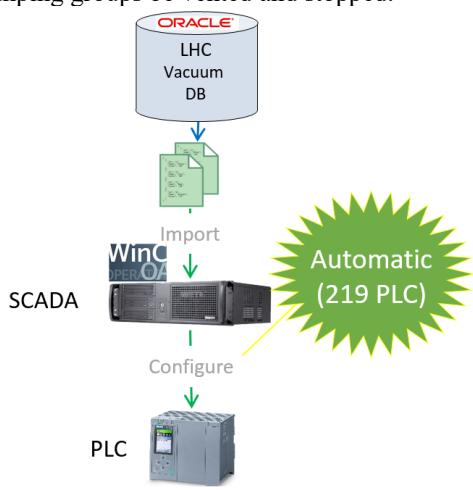


Figure 4: New strategy update procedure.

The new strategy delegates the update of parameters and names (defined in a database) to the supervisory application - making this procedure automatic.

As shown in Figure 4, the configuration step (i.e., the updating of names and parameters coming from the LHC

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

vacuum database) is included in the supervisory application update. This is the most common type of update required during Technical Stops. Technical Stops are already short (three-to-four days long), and the time allotted to the control software update is even shorter (half a day). The much faster update and the suppression of manual intervention errors are two obvious advantages. In addition, this procedure does not require for pumping to be stopped, nor does it interrupt the remote control and logging - also very important points.

Updates of the PLC code baseline still require a standard update procedure, but such intervention is rarely scheduled during Technical Stops.

New Ion Pump Controller and Sector Valve Interlock Animation

The ion pump controller measures the current of the power supply. This current is a function of the pressure in the vacuum vessel. The controller is therefore used to provide hardware interlocks to the sector valves with ion pump pressure level thresholds. The new ion pump controllers, installed during Long Shutdown 2, compliment a complete communication interface with remote control of all the parameters and access to all the statuses, including parameters and statuses of the interlock outputs.

To make use of this information, a new animated interlock hardware widget has been developed. Hardware interlock widgets are represented in the synoptic by an arrow pointing in the direction of the target valve. The arrow is represented either in green (no interlock) or in purple (valve interlocked). Sources of the interlock can be changed with the new remote control from one ion pump to another. As shown in Figure 5, the operator can now change the interlock schema on-the-fly and shift the interlock source from a faulty ion pump, to an intact one, and so maintain the level of protection even when a component of the interlock schema fails. This is not such a rare occurence in a vacuum system as large as the LHC's.

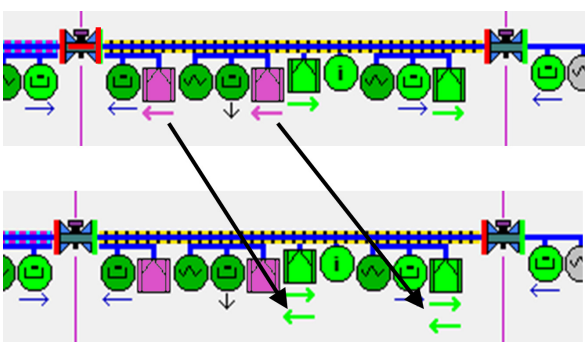


Figure 5: Hardware interlock animation in the synoptic.

I/O Gauge Device Type

Measuring and control cards for I/O gauge are installed in the arcs of the LHC tunnel (for both insulation and beam vacuum). Anticipating increased levels of radiation in the tunnel, the cards have been replaced with a radiation-resistance version. The software framework team took the opportunity to develop a new unified device type for all I/O gauges with a thorough protection scheme.

The Penning gauge is an on/off vacuum gauge to measure pressure ranges between $1x10^{-5}$ and $1x10^{-11}$ mbar. If this gauge were switched on or operated at pressures above approximatively $1x10^{-5}$ mbar - it may be damaged, or at least see its lifetime reduced. To avoid such damage, the gauge is protected either by the controller – or, in case of an I/O gauge – by the PLC. With the new I/O gauge device type, the protection schema has default parameters that can be changed on-the-fly from the supervisory application.

Figure 6 shows all of the protection parameters:

- "Protection Source Gauge": the third-party gauge used to protect the Penning gauge. This is the standard protection scheme that interlock/switch-off the Penning gauge in case of high pressure.
- "Self Protection OFF Threshold": same as above, but the source is the Penning gauge itself. It does not replace the above protection – it acts as a redundancy.
- "Self Protected Mode Threshold": above this pressure threshold, the gauge is switched on for only 16 seconds (duration required for a valid measurement) every 8 minutes. This advanced protection behaviour increases the lifetime of the Penning gauge.

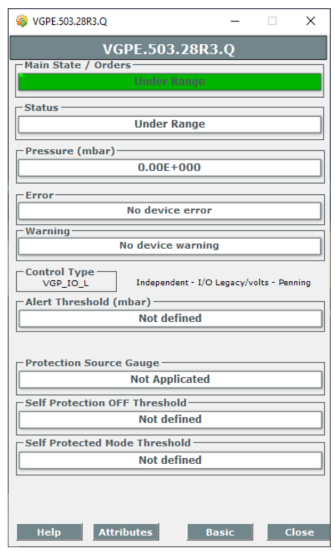


Figure 6: Details panel of an I/O Penning gauge.

NEW NOTIFICATION SYSTEM

The notification system sends e-mail/SMS message according to its criteria. The notification system already existed before Long Shutdown 2. The first version was working well and had already been used on a large scale, but it required advanced user skills and quite a long time to configure parameter criteria.

The new system offers predefined criteria according to the vacuum sector state. The default available modes are:

- Operation beam vacuum: the criteria are based on sector valve positions and the high vacuum gauge pressure value (threshold $1x10^{-7}$ mbar)

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV031

- Operation insulation vacuum: the criteria are based on the cryogenic alarm signal and rough vacuum gauge pressure value (threshold $1x10^{-3}$ mbar)
- On work/intervention: the criteria are based on the sector valve position and pumping group state (not pumping or disconnected)
- Bakeout/NEG Activation: the criteria are based on the sector valve position and pumping group states (not pumping or disconnected), the Bakeout controller state, and the gauge pressure value (threshold $1x10^{-2}$ mbar).

Additional predefined modes and their criteria can be easily defined on-the-fly by the software engineer in charge of the application, simply by adding ASCII text parameters in the supervisory application resources and configuration file. This file is parsed by the notification library during initialisation of the supervisory application client session.

ASSET MANAGEMENT

A functionality to directly manage assets [9] has been deployed to the supervisory application. It requires up-to-date synchronisation of the functional position names between the vacuum database and the CERN Enterprise Asset Management database. It offers an interface from the widget menu to create maintenance or repair requests. This functionality saves a lot of time and encourages proper equipment asset management. It has been largely used by the vacuum operators during the Long Shutdown period.

CONCLUSION

The Vacuum Supervisory application is one of the biggest industrial control applications at CERN with 1.5 million data points, more than 290 PLCs for fixed equipment, and more than 250 PLCs for mobile equipment. The main concern of the application upgrade has been device type standardisation. The layout changes for LHC Run 3 have been properly integrated in the supervisory application with significant changes in more than 40 vacuum sectors. The migration of the new controllers has been successfully integrated, taking the opportunity to increase functionality.

The new features and developments have increased the protection of the vacuum system, reduced effort required for its operation and monitoring, and improved diagnostic capability.

REFERENCES

[1] S. Blanchard *et al.*, "Vacuum pumping group controls based on PLC" in *Proc. PCaPAC'14*, Karlsruhe, Germany, Oct. 2014, paper WPO030, pp. 105-107.

[2] S. Blanchard *et al.*, "Bake-out mobile controls for large vacuum systems" in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC026, pp. 119-122.

[3] N. Chatzigeorgiou *et al.*, "Radiation tolerant conditioning electronics for vacuum measurements", in *Proc. TWEPP'18*, Antwerp, Belgium, Sep. 2018.

[4] E. Blanco *et al.*, "UNICOS Evolution: CPC Version 6", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper WEPKS006, pp. 786-789.

[5] O. Holme *et al.*, "The JCOP Framework", in *Proc.* ICALPECS'05, Geneva, Switzerland, Oct. 2005, paper O3_005.

[6] P. Gomes *et al.*, "Consolidations on the Vacuum Controls of the CERN Accelerators, During the First Long Shutdown of the LHC" in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper MOPGF104, pp. 322-325. doi:10.18429/JACoW-ICALEPCS2015-MOPGF104

[7] R. Ferreira *et al.*, "LTE/3G Based Wireless Communications for Remote Control and Monitoring of PLC-Controlled mobile vacuum device", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper MOCPL04, pp. 64-70. doi:10.18429/JACoW-ICALEPCS2017-MOCPL04

[8] S. Blanchard *et al.*, "Integration of Wireless Mobile Equipment in Supervisory Application", in *Proc. ICALEPCS'19*, New York, USA, Oct. 2019, paper WEPHA017, pp. 1102-1106. doi:10.18429/JACoW-ICALEPCS2019-WEPHA017

[9] A. Rocha *et al.*, "Integration of the Vacuum Scada with CERN's Enterprise Asset Management System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUPHA044, pp. 490-494. doi:10.18429/JACoW-ICALEPCS2017-TUPHA044

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CHALLENGES OF AUTOMATING THE PHOTOCATHODE FABRICATION PROCESS AT CERN

Cédric Charrondiere, Eric Chevallay, Thomas Zilliox, CERN, Geneva, Switzerland

Abstract

The CERN Photoemission Laboratory was founded in 1989 with the goal of studying laser-driven electron sources, for producing high-brightness electron beams within the framework of the Compact Linear Collider (CLIC) study. To produce these photocathodes, two processes run in parallel. The first process, which is slow and asynchronous, controls and monitors the evaporation of photoemissive material. For this first step several power supplies are controlled to evaporate different metals through the Joule effect, with the power maintained constant in time and the thickness deposited monitored. The second process is synchronized with a laser trigger ranging from 0.1 to 50Hz, where the photocurrent and laser energy are measured to calculate the Quantum Efficiency.

The control system for these processes has recently been renovated to benefit from the modularity of a PXI-based real-time environment using the standard CERN MiddleWare communication layer (CMW). This paper describes the challenges of the fabrication process as well as the flexibility introduced by using a PXI system.

INTRODUCTION

The CERN Photoemission Laboratory [1] was built with the goal to study and supply the CLIC Test Facilities (CTF 1 and 2) [2] with electron bunches of a few ps in time with a high electrical charge in the range up to 100 nC.

The CERN Photoemission Laboratory apparatus is divided in three parts:

- The Preparation Chamber where the fabrication of our photocathodes (the electron source) is done
- A DC Gun coupled with a beam measurement line for the characterisation of the cathode properties with a real, powerful, electron beam.
- A laser which is used to illuminate the electron sources during the fabrication and characterisation phases.

In the middle of the years 1990, after a few years of R&D, and the validation of the Cs-Te deposition technology, the laboratory received an important upgrade which enable the possibility to supply the CTF, our first client, with electron sources.

FABRICATION PROCESS

Physical Process Involved

The fabrication of the photocathode (Fig: 1) is the process of coating a substrate, the Cu photocathode plug, with a mixture of alkali metals and another element, tellurium or antimony. During this process the cathode is illuminated with a laser. A photoemission reaction occurs and produces an electron current, collected and measured to quantify the cathode's performance. The goal is to maximize the photocurrent and the quantum efficiency of the cathode while ensuring a very good source lifetime.

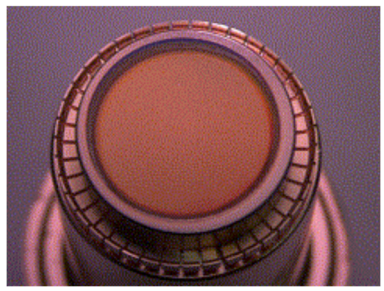


Figure 1: Cs-Te coating visible on the photocathode head.

In order to optimize the process, the system needs to be able to control several power supplies used to heat the evaporators and monitors the applied electrical power, temperature, laser energy, intensity produced by the photoelectron current, thickness of the coatings and vacuum pressure.

Instrumentation

- The Inficon XTM/2 deposition monitor is dedicated to monitor the alkali coating.
- The Inficon XTC/2 deposition controller is used to monitor the thickness of the tellurium coating.
- The Inficon XTC/3S is a spare deposition controller
- The TPG300 is a vacuum gauge controller to monitor the pressure in the vacuum vessel.
- The VIONIC 159 is the vacuum gauge controller for hot cathode gauge.
- The Heidenhain ND 780 is a positioning ruler, as a milling machine gives the position of the evaporator arm and the cathode arm in um.
- The Radiometer RM6600a is used as Joulemeter to monitor the laser energy when the light hits the cathode.
- Power supplies (Delta Elektronika + CERN made electronics) heat the evaporator (Cs or Te) by the Joule effect (control and instrumentation)
- Temperature monitoring with thermocouple and dedicated electronics
- Photocurrent acquisition with a precise laser synchronous timing system and fast sample/hold electronics.

CHALLENGES

Photocathode fabrication in laboratories like CERN is not reproducible because installations were designed for R&D and not for industrial production. Furthermore, with

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV032

each fabrication we often change parts of the process for learning purposes. Lots of parameters can affect the final result and we need to have a good knowledge and control of each of them:

- Deposition Flow Rate of Cs or Te (called Stoichiometric Ratio Control)
- Caesium availability in the evaporator
- Vacuum Quality
- Temperature (Alkalic Antimony Cathode only)
- Mechanical positioning of the Evaporators (ruler)

We learned how to upgrade the mechanical engineering aspects of the project to reach reproducible results and a fully automated process is now mandatory to assist the operator in controlling the process. All the tools are now available to start a real automatic process.

AUTOMATION

Architecture

The user should be able to monitor and control the main parameters with a refresh rate of approximately 1 Hz and plot them as a function of time. The entire process can take more than half a day and should not be stopped. Some intelligent devices like the laser energy meter or the thickness monitors communicate via GPIB or RS-232 The other devices are controlled by analog signals (between +/-10 V) and digital signals. To separate the GUI and the real-time process, a standard 3 tier design (Fig. 2) has been chosen. The communication between the real-time task and the GUI is done using the CERN Controls Middleware framework (CMW) [3]. Each equipment is abstracted by a dedicated software module.

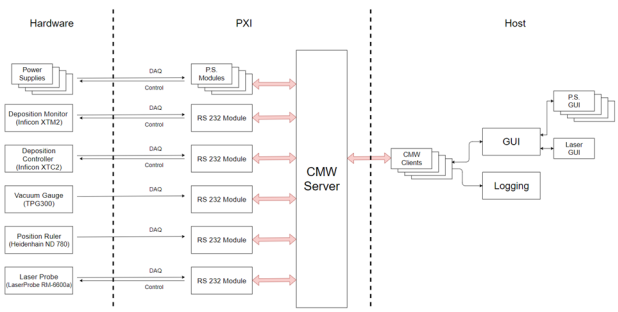


Figure 2: Automation software architecture.

The software is implemented using the Actor Framework [4], which abstracts the communication between the different modules, thereby increasing development efficiency. This framework is object-oriented, which allows us to add several layers of abstraction. Each serial device is a child of the RS-232 module class (Fig. 3).

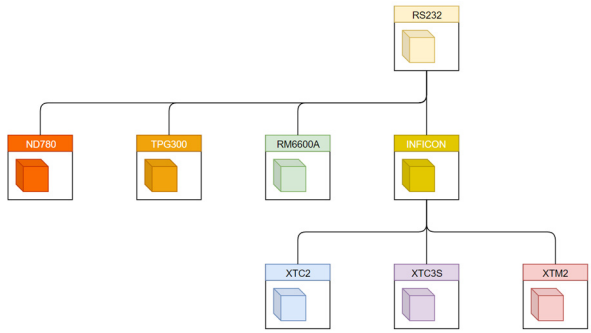


Figure 3: Class hierarchy of the RS-232 module.

Each serial device has maximum 3 features:

- Live streaming of the current data
- Getting the current settings
- Reacting to user commands

The RS-232 devices class abstracts all these features and has a common communication interface with the main module. The INFICON devices, being similar to each other, share a common parent. The addition of a new device and the maintenance of the current ones benefit from this implementation.

The most challenging part of the real-time software was the power supply implementation (Fig. 4).

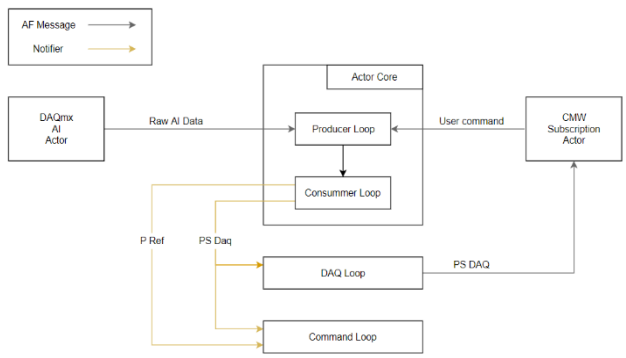


Figure 4: Power supply internal communication actor.

Each power supply is handmade at CERN and therefore has its own behaviour. To regulate the power supplies, an adaptive controller could also have been implemented but the idea was discarded due to a lack of time and the difficulty to make a system identification.

The actor framework's main drawback on LabVIEW real-time targets, is that each actor is re-entrant. This means they cannot be debugged using standard LabVIEW tools. To cope with this limitation, the RADE-logger [5] was heavily used to track errors. To be able to test the actor framework properly on real-time targets, additional support modules had to be developed, adding overhead compared to standard OOP LabVIEW implementations. In hindsight, using the Actor Framework on a real-time target might not have been the best design choice for this project, however it proved both scalable and adequate for the job.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Hardware Chosen for the Automation

The process doesn't require high speed acquisition or control, however it does require stability during a long run. To benefit from a Real-Time OS, a timing input and a robust CPU, the PXI controller NI PXIe-8821 with a PXIe-1082 chassis, was used. To communicate with the serial devices, a PXIe-8430/8 was used. To communicate with power supplies, a PXIe-4302 for the analog inputs, a PXIe-6738 for the analog outputs, and a PXIe-6535 for the digital inputs and outputs were used (Fig. 5).

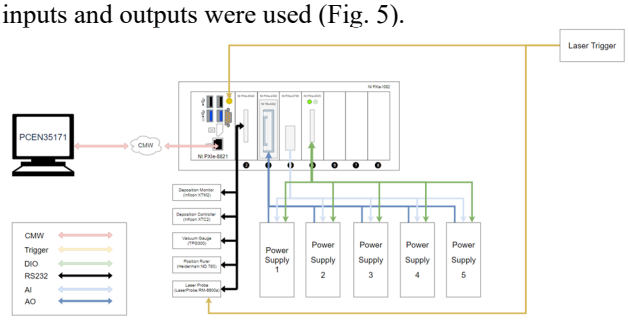


Figure 5: Automation hardware architecture.

Power Supply Control

In the application, the power supplies are controlled for power, but physically in the hardware, only the current parameter is controlled at the power supply. We use the voltage of the evaporator to compute the power of the evaporator as shown in Eq. (1).

$$P_{Evaporator} = U_{Evaporator} * I_{Power\ Supply} \qquad (1)$$

This power must be regulated in real time within a few hundreds of ms via the application. Up to five power supply values can be regulated in parallel.

Continuous Process

The power supply feedback is made with a 100ms cycle to comply with the control system design. Therefore, the analog and digitals inputs are acquired at 10 Hz. They following the power supply current, voltage and states and also the temperature of the fabrication bench as shown in Fig. 6.

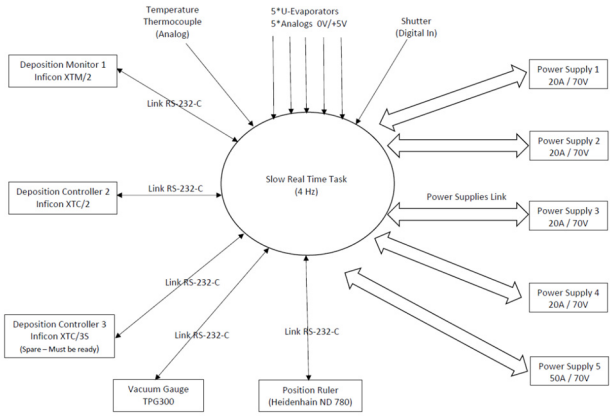


Figure 6: Slow process communication layer.

Several devices are connected to the control system using RS-232. As the fabrication of the photocathodes is a relatively slow process, it is sufficient for the feedback of the deposition of the different metals to be followed on the timescale of one second. The use of serial communication is therefore acceptable. The readings of the different devices are made at 4 Hz. 10 to 20 seconds are needed to get a proper deposition, and within this time we can pass from a good to a bad cathode. A precise quantity of matter needs to be deposited to maximize the quantum efficiency (QE), which is given by Eq. (2) [6].

$$QE = K\frac{Laser_{Energy}}{U_{Electrode}} \qquad (2)$$

Where K is a coefficient.

Triggered Process

The electrode voltage and the laser energy are monitored at 10Hz. This value allows us to compute the quantum efficiency that the operator will maximise to improve the quality of the photocathode. The acquisition of the radiometer, measuring the laser energy, and the electrode voltage are synchronized using the TTL trigger from the laser as shown in Figure 7.

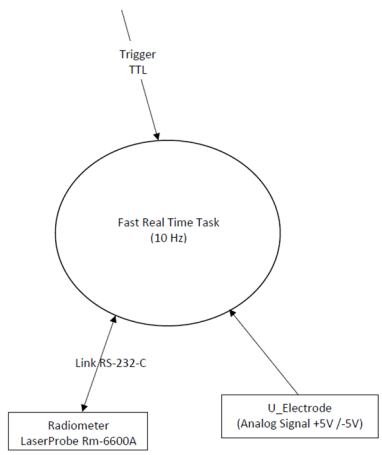


Figure 7: Fast process communication layer.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV032

RESULTS

To optimize the stoichiometric ratio of the Cs-Te (Cs2Te) alloy, you need to optimize the flow rate of the Caesium vapor. This optimization is a function of the (not constant) tellurium vapor flow and of the level of quantum efficiency. This is done by fine tuning the evaporator power.

This allows the QE to achieve record values greater than 20% (Fig. 8) but the process remains delicate. The key to achieve this is the monitoring of parameters such as photocurrent, laser energy, deposited thickness and a good control of the power.

The high quantum efficiency allows us to produce a cathode with longer lifetime (between weeks and months depending on the electrical charge produced and the frequency of use) addressing what is probably the main issue of these cathodes.

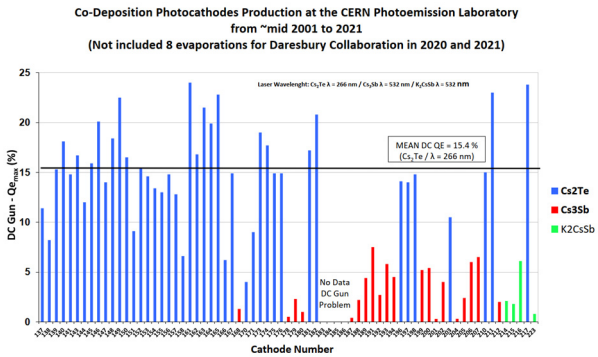


Figure 8: Quantum efficiency result.

Even if the process is not very fast, it requires both a flexible and deterministic control solution. The NI PXI systems provide both things. Moreover, the PXI platform opens the possibility to add additional cards and/or more mathematical processes in the future.

FUTURE IMPROVEMENTS

To improve the quantum efficiency, and further automate the solution, we need a deeper understanding of the physical process itself. This could be done via increased monitoring and perhaps, using machine learning, further optimising the process itself.

In general, adaptive controllers could be implemented after proper system identification. Interlock monitoring on the mechanical pieces could be added. The motors of the cathode holders and the vertical arms holding the evaporators could be controlled automatically. This will still require a deeper understanding of the manual process itself which today relies entirely on the experience of the operator.

CONCLUSION

Even though an operator is still required to monitor and control the photocathode fabrication process, the automation of various tasks such as the evaporator control, the synchronization of each device and the reproducibility introduced by this new implementation allows CERN to be technologically ready for full automation of this process in the future. In addition, the accuracy of the power regulation as well as the availability of the data gives us more flexibility in the automation.

REFERENCES

[1] E.Chevallay, J.Durand, S.Hutchins, G.Suberlucq, M.Wurgel, "Photocathodes tested in the dc gun of the CERN photoemission laboratory" http://dx.doi.org/10.1016/0168-9002(94)91293-9

[2] G. Suberlucq, "Technological challenges for high brightness photo-injector", in *Proc. EPAC'04*, Lucerne, Switzerland, Jul. 2004, pp. 64-68.

[3] CERN-AB-2003-104, The controls middleware (CMW) at CERNstatus and usage

[4] A.C. Smith, S.R.Mercer, "Using the Actor Framework in Labview", https://forums.ni.com/ni/attachments/ni/7301/130/1/Us-ing%20the%20Actor%20Framework%203.0%20in%20LabVIEW.pdf

[5] O. Ø. Andreassen, D. Kudryavtsev, A. Raimondo, A. Rijllart, V. Shaipov, R. Sorokoletov "The LabVIEW RADE framework distributed architecture", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper WEMAU003, pp. 658-661.

[6] E.Chevallay, CTF3-Note-104, EDMS 1215630, "Experimental Results at the C.E.R.N. Pho-toemission Laboratory with Co-deposition Photocathodes in the Frame of the CLIC Studies.", https://clic-study.web.cern.ch/sites/clic-study.web.cern.ch/files/pdfs/notes/CTF3Note104.pdf

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

DISTRIBUTED TRANSACTIONS IN CERN'S ACCELERATOR CONTROL SYSTEM

F. Hoguin, S. Deghaye, P. Mantion, J. Lauener, R. Gorbonosov, CERN, Geneva, Switzerland

Abstract

Devices in CERN's accelerator complex are controlled through individual requests, which change settings atomically on single Devices. Individual Devices are therefore controlled transactionally. Operators often need to apply a set of changes which affect multiple devices. This is achieved by sending requests in parallel, in a minimum amount of time. However, if a request fails, the Control system ends up in an undefined state, and recovering is a time-consuming task. Furthermore, the lack of synchronisation in the application of new settings may lead to the degradation of the beam characteristics, because of settings being partially applied. To address these issues, a protocol was developed to support distributed transactions and commit synchronisation in the CERN Control system, which was then implemented in CERN's real-time frameworks. We describe what this protocol intends to solve and its limitations. We also delve into the real-time framework implementation and how developers can benefit from the 2-phase commit to leverage hardware features such as double buffering, and from the commit synchronisation allowing settings to be changed safely while the accelerator is operational.

DISTRIBUTED TRANSACTIONS: INTRODUCTION

In CERN's accelerator control system, clients address devices individually with a request made on a property of a device. To modify a property across multiple devices, requires as many requests as there are devices. Until all the requests are made, an accelerator is in an undefined state. Moreover, if any of the requests fail, additional actions are required to understand why it failed and how to fix it, leaving the accelerator in an undefined state for a potentially large amount of time.

In order to solve this problem, distributed transactions were designed and implemented in CERN's accelerator control system.

GOALS, LIMITATIONS, AND DESIGN OF DISTRIBUTED TRANSACTIONS

The goal of distributed transactions is to ensure that a set of modifications occurring on distributed nodes are either completely recorded, or not at all. A node can be anything, but the first example that comes to mind is a database. This behaviour is typically achieved by using a two-phase commit. The generic workflow is as follows:

- A transaction is opened with the different participating nodes;
- Modifications are made on the different nodes in any order and over an unspecified period of time;
- The commit then occurs in two steps:
  1. A commit is sent to all the nodes which perform the required checks and ensure that locally there are no errors;
  2. If no errors are reported by the first commit, a second one, confirming the wish to commit is sent. From then onwards, the modifications are permanently recorded;
  3. If, on the other hand, errors are reported during the first-phase commit, the transaction is rolled back on all the nodes and no modifications are recorded.

DISTRIBUTED TRANSACTIONS IN ACCELERATOR CONTROLS

In the context of CERN's accelerator controls, the nodes are the low-level equipment controllers, the so-called Front-End Computers (FECs). The modifications are changes of the underlying equipment settings. A nominal transaction for CERN accelerator controls contains the following steps:

- The client opens a transaction with a unique transaction ID on all the devices it plans to modify, via a synchronous middleware call to a standardised property;
- The client modifies the settings via a synchronous middleware call. Again, the order and period of time are unspecified;
- The client asks the devices to test their new settings via a synchronous middleware call to a standardised property (TRASACTION.TEST);
- If all the devices report a successful test, the client commands the timing system to broadcast a commit event for the transaction. Upon reception, the FECs permanently commit the new settings;
- Otherwise, if one or more devices report an error during the test, the client asks the timing system to broadcast a roll-back event which will trigger the discard of the settings set with the corresponding transaction ID.

In the workflow described above, several implementation choices are already visible. One key element is the way to transport the final commit and roll-back. CERN's timing system, whose purpose is, among other things, to distribute events to the whole accelerator complex, is used to trigger a commit or rollback of a transaction. Even though the transactions are not meant to synchronise the moments at which the commits are performed, by using the timing system to transport these two events, we combine the possibilities offered by a global commit event with the synchronisation possibilities of the timing system.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV033

In addition, mainly for test purposes, the client may trigger a commit or rollback event without using the timing system. This is done by telling every device in the transaction to commit or rollback via another middleware call to the TRANSACTION.COMMIT or TRANSACTION.ROLLBACK properties. It is acknowledged that in this case, the commit loses its atomicity, since the client is limited to regular TCP connections to do this, without any multicast possibility.

It should be highlighted that, on the low-level equipment controllers (FECs), a single server typically handles many devices. Clients do not use this information in any way and must treat each device regardless of the FEC they are handled by. However, this has some implications on the design of the server-side commit process. The implementation of the transaction protocol in a device server, where a transaction is processed as a whole entity (in opposition to device by device), is explained in the following sections. Why such a design choice was made, its constraints, and how the technical challenges were solved are also described.

DISTRIBUTED TRANSACTIONS IMPLEMENTATION IN FESA

The Front-End Software Architecture (FESA) framework is used to do real-time programming to control accelerator hardware in CERN's control system. It offers the possibility to react to specific events through an API giving access to all the concerned devices at once. Many systems developed with FESA are a composition of multiple devices, and if settings are changed on multiple devices within a transaction, this must be seen as one change of all the settings, rather than multiple changes of individual devices.

Since clients are not aware of this (they always treat devices as individual entities), this must be transparently implemented in FESA. In addition, a FESA device server can be run either as a single process (see Fig. 1) or as two separate processes, called server process and real-time process, which interact through a shared memory and several message queues (see Fig. 2).

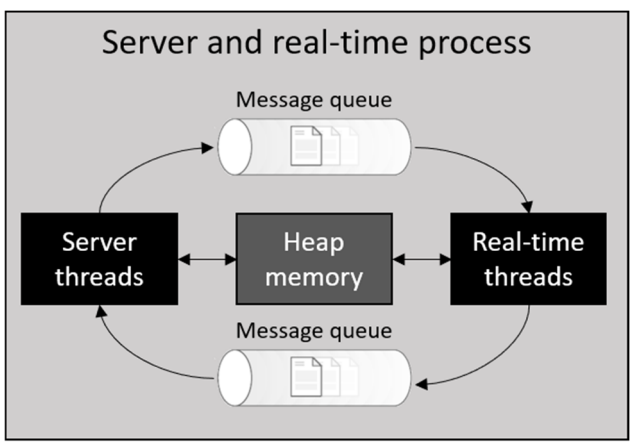


Figure 1: FESA server running as a single process.

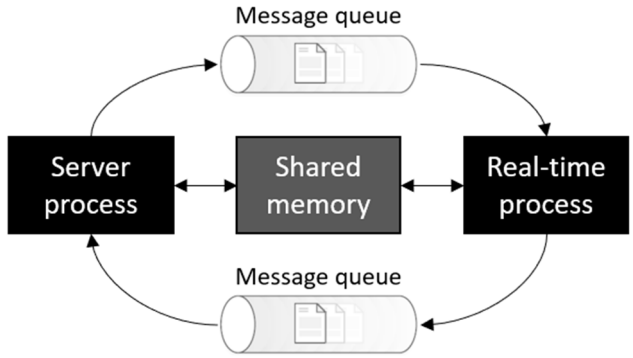


Figure 2: FESA server running as two processes.

This has several implications:

- FESA must be able to uniquely identify and pass the data from a transaction across processes.
- Since the transaction is opened sequentially on all the devices by the client, FESA cannot know in advance the size of the settings that will be modified in the transaction.
- When a transaction is tested by the client, test requests will happen sequentially on every device. FESA must test the full transaction on the first received test request and ignore subsequent test requests on other devices.

Many failures can happen during a distributed transaction (hardware error, device server crash, network issue, timing system issue). A commit event is thus not guaranteed to be processed by all the devices in the transaction. Because of this, the processing of the commit event by the devices should be as small an operation as possible, e.g., a buffer swap. Devices must use the test request to prepare for an upcoming commit or rollback. Consequently, the implementation does not guarantee the state of the system in case of a failure during the commit. Furthermore, due to the nature of the timing system, it is difficult to provide feedback to the client in such a case. Some solutions exist to mitigate this problem [1], at the cost of an increased complexity.

Finally, in order to provide a transaction implementation across as many devices as possible, it was decided that FESA would offer two modes:

- A transparent mode, with no code modification required by the users, but without the possibility to leverage some hardware capabilities such as double buffering. This is intended for cases where validation of settings is done entirely in software, which guarantees that commit events will successfully apply the new settings unless an unpredictable hardware error occurs.
- An advanced mode, which requires the user to implement entry points for test, commit, and rollback. In this mode, the test event can be used to prepare the hardware, for example by preparing a

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

buffer containing the new settings to apply, and the commit event can be used to trigger a buffer swap, applying the new settings atomically in a very short time.

These two modes are mutually exclusive in a FESA process.

TRANSACTIONS IMPLEMENTATION IN FESA: TRANSPARENT MODE

In order to fully understand the implementation of transparent transactions, it is first necessary to understand how a normal modification of a setting works (outside of a transaction):

- The client sends a new setting.
- Its validity is checked by some user code, which may accept or reject the new value.
- If the new setting is valid, the memory containing the setting values is updated with the new value [2]. If the setting needs to be applied immediately, an event is sent to the real-time FESA process, possibly across process boundaries if running in split mode.
- The event is received and processed by the real-time part, which applies the new setting on the hardware.

If the setting is instead modified within a transaction, the sequence is as follows:

- The client sends a new setting.
- Its validity is checked by some user code.
- If the new setting is valid, its value is saved in a buffer dedicated to this transaction. If the setting needs to be applied as soon as the transaction is committed, an event is prepared and saved in the transaction buffer.
- The client sends other settings, and the same process is repeated until a commit or rollback event happens (in transparent mode, the test event is ignored, as the validity of settings has been checked already).
- If a commit event happens, the memory containing the setting values is updated with the buffered setting values. All the buffered events are triggered.
- The events are received and processed by the real-time part, which applies the new settings on the hardware.

Thus, in transparent mode, the modification of settings within a transaction are equivalent to modifying all the settings in rapid sequence.

While the transparent mode offers the main advantage of being usable without user code modification, it does not cover one key element of the nominal sequence which is the possibility for Sets to be done in an unspecified order. To support this, a more sophisticated implementation is required, the advanced mode.

TRANSACTION IMPLEMENTATION IN FESA: ADVANCED MODE

The goal of advanced mode is to allow the user to benefit from the 2-phase commit (test, then commit) to test new settings only after they have all been set, and to leverage the capacity of the controlled hardware to prepare for a commit.

A fictitious but simple example (which can be applied to many other use cases) can demonstrate why this is important:

Consider a lab-bench power supply, which has a power output limit of 300 watts. It is connected to a FESA process which can modify its voltage and maximum current through a serial connection.

When a new voltage or maximum current is requested by a client, the user code must check that the product of the two does not exceed 300 watts. Assume that it is currently configured to 5V and a maximum of 20A.

If a client wants to set it to 30V and 5A, without the transactions, it must be careful to first set the maximum current, otherwise the maximum power will exceed 300W in the transition (30V times 20A). This requires the client to know about this logic, which is not feasible in the case of automated settings management.

The transaction implementation in advanced mode solves this problem by delaying the validation of the settings until the test request happens. The user code can still reject a setting value as soon as it is sent if it is invalid on its own (e.g., setting a voltage of 10000V), and in addition, it can check the validity of the settings combination in the code that is called when a test event happens. The API provided by FESA in that case transparently provides the values stored in the transaction buffer if present, or previously known otherwise.

Now also assume that the power supply can receive its new voltage and maximum current settings but wait for a further command to apply them. In addition to ensuring the settings validity, the implementation of the test event would then send the new values to the power supply. If anything goes wrong at this point, it is still possible to signal a test failure to the client, who can roll back the transaction, or try with other setting values.

Finally, when the commit event arrives, the only thing left to do is to tell the power supply to apply the new settings. The possible failure surface has thereby been reduced to the strict minimum, allowing the clients to cancel the transaction if anything else goes wrong.

While this example is voluntarily simple, it applies to a lot of hardware used in accelerators, such as power converters, digital oscilloscopes, and much more.

ADVANCED MODE OUTSIDE OF A TRANSACTION

As explained above, transactions in advanced mode require the user to adapt their code to handle the test, commit and rollback events. However, the settings can still be changed outside of a transaction, in which case they need to be applied immediately.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV033

For such cases, FESA simulates a transaction: the request to modify a setting will cause FESA to generate a test event, and if it is successful, a commit event. If the test event fails, the new setting value will not be applied, and the client will receive the test event error message, informing why the new setting could not be applied.

While this approach may seem complex at first, it has many advantages: the complexity remains in the FESA framework, where it is thoroughly tested. The user only has to write a single implementation of the code which communicates with the hardware, and as a result only has to test one code path.

TRANSACTIONS ACROSS PROCESS BOUNDARIES

The FESA framework may be used in what is called "split mode". This is meant for highly critical systems, which should always be up and running. In this mode, the real-time part of a FESA deployment is launched in a process, while the server part, responsible for handling client requests, runs in a separate process. This can be seen as a way to sandbox the real-time process.

The two processes communicate using standard IPC methods: shared memory, message queues, shared mutexes and condition variables. From a user point of view, nothing changes feature-wise, but there are a few restrictions, mainly due to the fact that pointers are not valid across process boundaries.

In order to implement the transactions feature in FESA, these considerations had to be taken into account. As a result, the transaction buffer, allocated when a transaction is opened, is a dynamic structure which cannot contain pointers. The devices and their settings are given unique indices, allowing to identify them across the two processes of a split mode deployment. Every transaction buffer also contains a mutex and a condition variable, which are used to signal the completion of the test, commit, and rollback events.

There is another consequence of running in split mode: the memory allocated for the transaction buffer must be contiguous, as it is mapped between the two FESA processes using the *mmap* system call, which takes an address and a size. However, it is not possible to know in advance how much memory will be needed for the transaction buffer: the client opens the transaction sequentially on an unpredictable number of devices. Furthermore, it is allowed to add devices to the transaction as long as the test request has not been sent.

To solve this problem, a naive solution would be to allocate all the memory potentially needed immediately, by computing the maximum size of the transaction, which corresponds to the size of all the settings of all the devices in the server, plus the size required for the transaction's bookkeeping structure. However, many FESA processes operate on machines which have a low amount of memory, without enough memory left to do this. This means that it would be impossible to conduct a transaction on such a server, even if it would affect a single device.

The solution is to take advantage of the MMU of the CPU, and the capabilities of the operating system: it is possible to reserve a chunk of contiguous memory in a process, without actually mapping it to any physical memory. This works even with memory shared between processes, by using the appropriate flags in the call to *mmap*: with PROT_NONE as protection together with the MAP_PRIVATE and MAP_ANONYMOUS flags, the memory is marked as inaccessible, causing the operating system to not allocate any physical memory, but to still reserve the address space in the process, guaranteeing a contiguous memory block. When the buffer needs to be extended, and some actual physical memory allocated, *mmap* is called again, allowing to allocate an arbitrary amount of the reserved memory. This time, we pass PROT_READ and PROT_WRITE to make the memory readable and writable, together with the flags MAP_FIXED and MAP_SHARED, ensuring respectively that the address of the reserved memory is not changed, and the memory is visible in other processes.

Tests on multiple Linux platforms have shown that this works as expected. On Windows platforms, the same effect can be obtained by calling *VirtualAlloc* with the flag MEM_RESERVE, and then MEM_COMMIT.

CONCLUSION

The design and implementation of distributed transactions in CERN's accelerator control system have been described. During the design process and for simplicity reasons, a choice was made to not make it resilient to hardware errors, but instead to allow the detection of errors until as late as possible before the commit happens. The reason is that, unlike a database, if a hardware error occurs, rolling back to the previous state is not necessarily possible, nor desirable.

Because of this, implementing a strong guarantee on the atomicity of transaction commits is not possible. Any attempt at doing so would be much more complex than the presented implementation and would require compromising on other aspects, such as the time needed to process a commit. In addition, keeping the distributed transaction design relatively simple ensures that it is possible to implement on any device. In practice, this allows to offer in FESA both a "transparent mode", allowing users to enable transactions on their devices without having to write a single line of code, and an "advanced mode", where users can fully benefit from the flexibility of the 2-phase commit.

Nevertheless, given the nature of the commit and rollback events, a logical next step is to implement a system of feedback to allow clients to know immediately if a commit failed, and why. Since the commit and rollback events are broadcast by the timing system, this requires an extension of the protocol to allow clients to be notified when that happens.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] P. Bailis, A. Davidson, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Highly available transactions: Virtues and limitations," in *Proc. VLDB'14*, Hangzhou, China, September 2014, pp. 181-192.

[2] F. Hoguin and S. Deghaye, "Solving the Synchronization Problem in Multi-Core Embedded Real-Time Systems", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 942–946, doi:10.18429/JACoW-ICALEPCS2015-WEPGF102

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV034

DEVELOPMENT OF AN AUTOMATED HIGH TEMPERATURE SUPER-CONDUCTOR COIL WINDING MACHINE AT CERN

H. Reymond, M. Dam, H. Felice, A. Haziot, P. Jankowski, P. Koziol, T.H. Nes, F.O. Pincot, S.C. Richter, CERN, Geneva, Switzerland

Abstract

Within the framework of technology studies on future accelerators, CERN has initiated a five-year R&D project aimed at the evaluation of REBCO (Rare Earth Barium Copper Oxide) High Temperature Superconductors (HTS). The study covers a number of areas from material science to electromechanical properties. The REBCO high-field tape will be tested on different HTS magnet prototypes, such as HDMS (HTS Demonstrator Magnet for Space), GaToroid (hadron therapy Gantry based on a toroidal magnetic field) and other smaller coils that will be fabricated to study the tape's potential. To assemble the HTS coils, a new automatic winding station has been designed and constructed at CERN. A touch panel combined with embedded controller, running software developed in-house provides a sophisticated, yet intuitive and user-friendly system aimed at maintaining perfect coil winding conditions. In this paper, we describe the mechanical choices and techniques used to control the seven HTS spool tapes and the winding machine. We also present the analysis of several coils already produced.

INTRODUCTION

Whether for the next generation of high energy particle accelerators, or for the development of new equipment dedicated to physics or medical research, one of the main decisive factors in the future will be the industry's ability to produce superconducting magnets with a field of at least 20 T [1].

Being one of the world leaders on superconducting magnet technology, CERN has been given responsibility of work-package 10 (WP10) "Future Magnets" as part of their contribution to the EuCARD-2 project [2], supported by the European Commission's Seventh Framework Programme (FP7). WP10's main goal was to manufacture and qualify an HTS cable, within real demonstrator coils and magnets, having useful characteristics for accelerator magnets (dipole field of 20 T, industrialized production, affordable cable, …).

After studying several possible candidates, the REBCO tape was chosen as an appropriate candidate, mainly because of its mechanical properties, but also due to its availability through many different suppliers, and finally because it doesn't require any further treatment before assembly [3].

In 2017, the ARIES European program was launched with the objective of improving the REBCO tape current density [4]. In parallel, CERN initiated a 10-year research program (CERN HTS program) to study and develop dipole magnets with magnetic fields beyond 20 T [5].

The Prototypes and Demonstrators

To test and evaluate the new requirements, several R&D studies were launched, each focusing on different technologies.

- The HTS demonstrator magnet for space (HDMS) aims at validating the feasibility of a new generation of magnet to be used for a spatial spectrometer [6].
- The 20 T HTS Clover Leaf End Coils magnet to study the mechanical and magnetic aspects of so-called overpass / underpass coil end assembly [7].
- The 8.2 T toroidal coils, which would be the basis of the GaToroid CERN proposal, for a new gantry design in the field of Hadron cancer treatment [8].
- The small coils program as a general study of high-field accelerator magnets for Hadrons and Muons (solenoid, undulator) [9].

The Coil Winding Machine

In order to co-wind stacks of REBCO tapes, dedicated tools and custom machinery is needed for its assembly. It was therefore decided that the coil-winding machine would be built in-house, at CERN. As a result, different teams involved in anything from magnets design, mechanical studies to control system development, joined forces to manufacture and assemble the machine in just a few months [10].

THE WINDING MACHINE DESIGN

One of the main requirements of the machine design was the semi-automation of the winding process to be capable of working with up to 7 tapes simultaneously. An important request was the ability to set up and control in a closed loop the tension on the spools, with added monitoring and storage of the results.

The winding station was built as a long table made of aluminium profiles with 7 spools, each connected via a clutch to a dedicated motor (Fig. 1). Next to the spool holder, custom-made tape routing equipment was installed, both guiding and measuring the tension applied to the tapes. In addition, tape alignment tooling with a built-in length encoder was incorporated into the machine to merge and adjust the tape position before winding the coil.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV034

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

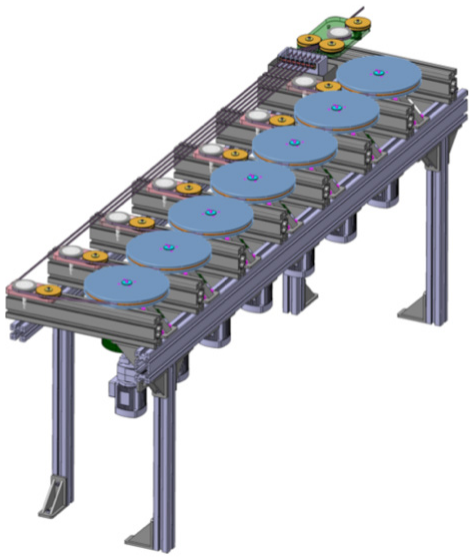


Figure 1: Winding station 3D model.

THE CONTROL SYSTEM

After assembling the winding station, the decision was made to build and develop a control system using a CompactRIO [11] controller from National Instruments, taking advantage of its combined RT-FPGA nature, and also because some machines hosted in the workshop had already been successfully renewed with this solution. To allow easy and intuitive human-machine interfacing, a Windows based touch panel was added to the system, allowing the user to constantly monitor the winding process parameters, as well as reconfigure regulation coefficients.

Early in the control system design phase we realized that even minor abnormalities in the operation might have critical impact on the components and the winding process product itself. It was clear that maintaining perfect tension conditions (7 – 100 N) of REBCO tapes would be key to the performance of the manufactured coil.

During the initial design, three important aspects were identified. First, the controller is supplied with quarter bridge strain gage inputs and AO voltage modules. Second, gage signals are used as process variables of the PID regulation loops, running on the embedded FPGA, while the outputs operate as a source (0 – 10 V range) to control the clutches. Third, the cRIO controller also handles DIO signals, connected to the motor relays, safety system buttons and encoders (Fig. 2).

A logging option was added to keep track of parameter changes. These are stored in a file referenced to the manufactured coil for future quality insurance assessment.

Mechanical Control of the Spools

When the machine is running, only the clutch is actively controlled, while both the gear and the motor are statically configured. This makes it possible to control the final torque, speed and cable tension by regulating the clutch. Due to safety-related aspects, the station is designed to disengage power transmission when idle. The clutch is controlled proportionally with a current between 0 and 2A, allowing active control of the spool.

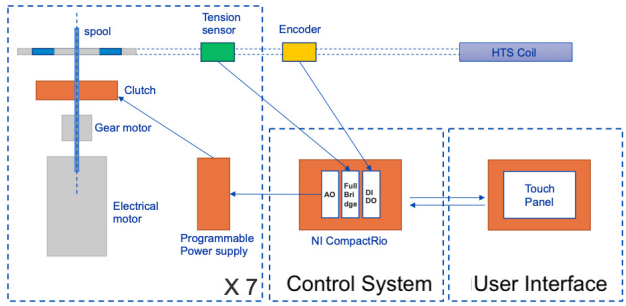


Figure 2: Mechanical diagram of the machine.

Sensors

One important factor when choosing suitable tension sensors is their readout range. Those selected for the assembly have slightly higher limits than required (200N), but one must consider their location in reference to the spools. As the point of contact between the cable and the strain gage pulley is rotated by 45° relative to the pull direction, the real maximum value of the tension that can be read on the tape is closer to 140N. This value can be adapted by orienting the sensor differently.

The Embedded Controller

The cRIO-9053 controller used for the project has proven to be a good choice in terms of its price/capability ratio. Our optimizations, such as pipelining and reduction of critical paths, allowed us to implement the software in a relatively small FPGA.

Software Architecture

Since the final implementation of the winding machine was not completely defined when the project started, we had to choose a flexible yet scalable software architecture that could easily be adapted to changing project requirements.

The software was developed using an optimized QMH (Queued Message Handler) software architecture. By offloading most resource-consuming tasks such as PID calculation to the FPGA, and by running the GUI on an external Windows based touchscreen-enabled computer, we managed to fit the application on the RT controller.

The FPGA Application

An FPGA-based firmware executes the most critical and time-consuming calculations related to the PID algorithm as well as direct hardware interfacing. Of seven parallel processes, three of them are related to regulation. Even if coil winding is a relatively slow process, the stable clocks of the FPGA tasks allow them to react very quickly to any change in the tape's tension by running the main loop with a cycle time of a few milliseconds.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV034

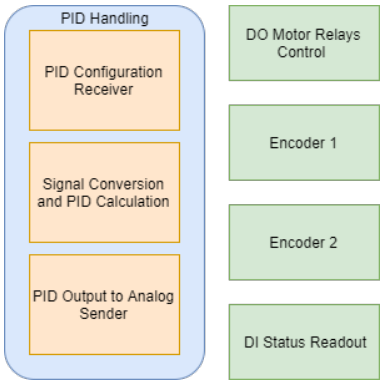


Figure 3: FPGA application parallel processes.

The FPGA implementation is divided into 3 main processes (Fig. 3). The First process is mainly dedicated to configuration management and parameter updates. The second process performs low-level signal calibration and conversions and stores calculated PID outputs into memory blocks. Finally, the third process reads control variables from memory and writes to the analogue and digital output modules.

Thanks to this modularized approach, the code is compartmentalized and simplified, preserving high configurability (fragments of program logic can be disabled or supplied with simulated parameters instead of real ones).

GUI and User Interaction

The main challenge when implementing a graphical user interface for the project was to keep it simple enough to ensure clear understanding of the machine status, yet deliver a fully functional tool to the operators. This was made even more difficult considering the low resolution and relatively small size of the touch-panel (Fig. 4). Ultimately, close cooperation with the operators and an iterative approach to development paid off with satisfactory results. The user interface is split into three panels (Fig. 5): Operator settings, runtime monitoring view and expert settings.

Figure 4: Winding machine with touch-panel user HMI.

The first panel consists of elementary winding process configuration, where the user needs only to specify tapes and tension to be used, in addition to optional informational values like operator or coil name.

The second tab provides run-time information and controls. In the central part, the application plots the current tension on a graph while on the left there is an adjustable set point indicator. This way, the user can perform slight tension adjustments during the winding process. The lower part of the GUI is reserved for the encoder readings (they are used to track used tape length and coil layers achieved). Two registers are kept for each encoder: temporary and total reading. The former resets on operator request and the latter is cleared only on winding process start-up. From this panel, the operator can start and stop regulation.

The “Expert Settings” panel is meant to be used only by experienced engineers to modify PID, sensor calibration and debug functionality. This tab is also password protected as because using these options by non-qualified users could damage the coil and REBCO tapes. A final feature from this panel is the live monitoring of some key electrical elements, like the power supplies, the safety stop relay and the motor contactors.

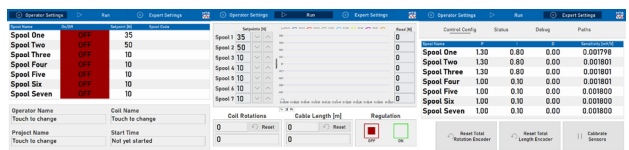


Figure 5: Application GUI.

PID Regulation

The coefficients of the PID regulations were first estimated, based on a theoretical approach. Then by using a spool of a few meters of copper tape, we adjusted them by executing dozens of tests of the regulation loop. During this process, some mechanical changes were made for the position of the strain gages and the pulleys. Following the final acceptance of the mechanical assembly of the machine, a test campaign was executed with 7 spools of copper tape during which the complete regulation loop was fine-tuned (mechanical, electrical and software parts).

COIL PRODUCTION RESULTS

Since 2020, the winding machine was used to produce HTS small solenoid, Undulator, HDMS, and Gatoroid coil prototypes. The currently produced pieces have been evaluated and mechanical parameters were assessed to improve the winding processes. Many trials were performed with partial winding of these coils and iterations of the assembly protocol, conducted to validate most of the coil prototypes production steps.

The Solenoid Coil

The first prototype (Fig. 6) was made using copper tape. It allowed us to validate some steps of the coil assembly:

- The tape tension (30 N)

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV034

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- The soldering process of the first turn on the copper ring support (before the winding)
- Soldering paste and heater device (T = 176° C) between each turn

Then the next prototypes were wound on another machine, as the seven spools winding machine was occupied by other projects (HDMS/Gatoroid).

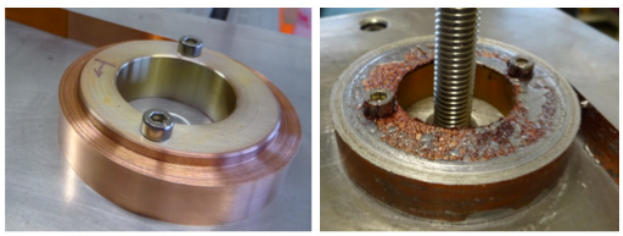


Figure 6: a) Solenoid coil winding, b) Soldering paste deposit.

The Undulator Coil

The winding machine was used to produce the very first vertical racetrack (VR) coil prototype [12], with two 4 mm copper tapes (Fig. 7). The simultaneous winding of two tapes was validated, but the distribution of the solder in between tapes was not uniformly distributed as needed. It was decided to wind the following prototypes in a vertical winding orientation, which was possible on another winding machine already present in the workshop, which was adapted for this purpose.

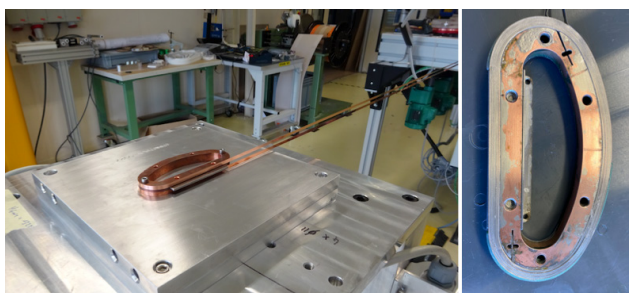


Figure 7: a) Winding of the first VR coil, b) The final coil after the soldering process.

The HDMS Coil

Five coils were made with the winding machine. Only the third and the fifth did not have geometrical errors. Finally, only the fifth one was retained to validate the winding process, using three copper tapes 12 x 01 mm, with an applied strain of 20 N on each. Several steps of the HDMS coil assembly (Fig. 8) still need to be tested (last turn fixing method, …) or studied (external copper ring to maintain the coil) but this is beyond the scope of the winding machine.

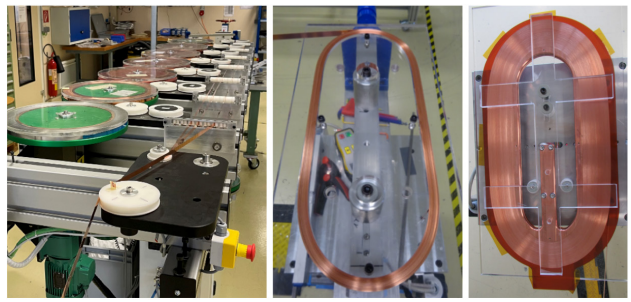


Figure 8: a) The winding machine with the 3 tapes, b) The HDMS coil winding, c) The complete coil.

The Gatoroid Coil

A first trial was done with this assembly scheme: six copper spools installed on the winding machine (1 x 0.5 mm, 4 x 0.1 mm, 1 x 0.5 mm). Several winding iterations permitted to get the optimal strain to apply on the tapes (50 N on 0.5 mm, 30 N on 0.1 mm). A second trial, using an S-2 fiberglass sleeve, was done to validate the insulation between the layers (Fig. 9). This also confirmed the tension and trench geometry for the coil support. A new mount for the fiberglass sleeve was added to remove friction at the entry point. The assembly process of the second layer spools was also validated by adding a second winding station, made of one axis and six spools. The six tapes were passed through the insulations and were again separated into six independent spools in the winding machine.

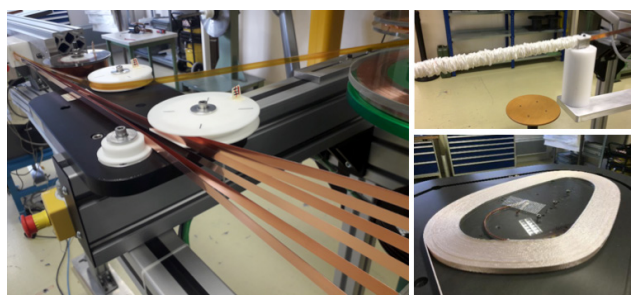


Figure 9: a) The winding machine with the 6 tapes, b) The S-2 Fiberglass stored on the tapes path, c) The Gatoroid coil.

CONCLUSION

The production of the first batches of coils showed that the winding machine works properly. Considering our broad experience with the NI ecosystem, time-constraints, and the robustness of FPGA based solutions, a CompactRIO controller was a clear choice as a basis for the control system. The complete control loop, from the strain sensors, the FPGA running the PIDs, up to the clutches is now set up with very good accuracy. Some mechanical parts have been reengineered during the test and validation campaign (position of pulleys in the merged tapes area, space between pulleys, orientation of the strain sensors). The GUI has been improved to offer more features during operation (easy control settings from the RUN panel, logging of all the commands for future support cases investigation). The winding machine now provides

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV034

an efficient service for the production of HTS coil prototypes at CERN. The possibility to wind and unwind a coil gives freedom in using copper tapes (less expensive), for process validation purposes.

FUTURE PLANS

The validation and initial tests of some winding processes have uncovered needs for mechanical improvements. Several new modules are being designed for some specific phases of the coil winding, such as new fiberglass sleeve storage and mounting system, a thermoforming device for the polyimide insulation thin film layer, and a new dimensional measurement control system for the respooling phase.

On the control and software side of the winding machine, some new features are already planned for better management of the HTS tapes inventory. The idea is to link the GUI with a database to get tape identification and parameters. This will be done by scanning a QR code on the spools to then import automatically the associated properties from the database. This will improve the QA and traceability of the produced coils, by saving tape characteristics with the data generated during the assembly. At the end of the process, this data will be archived in the database. The lengths of used tapes will also be automatically updated to allow an accurate management of available stocks.

REFERENCES

[1] X. Wang, S.A. Gourlay, S.O. Prestmon, LBNL, USA, "Dipole Magnets above 20 Tesla: Research Needs for a Path via High-Temperature Superconducting REBCO Conductors", Instruments, Nov. 2019. doi:10.3390/instruments3040062

[2] L. Rossi, A. Badel, M. Bajko, A. Ballarino et al. "The EuCARD-2 Future Magnets European Collaboration for Accelerator-Quality HTS Magnets", IEEE Transactions on Applied Superconductivity, Dec. 2014. doi:10.1109/tasc.2014.2364215

[3] L. Bottura, CERN, Switzerland, "EuCARD-2, Milestone Report 68, Technical and Economical Comparison YBCO/Bi-2212 Magnets".

[4] L. Rossi, C. Senatore, "HTS Accelerator Magnet and Conductor Development in Europe", Instruments 5 (2021). doi:10.3390/instruments5010008

[5] G. de Rijk, "CERN R&D for HTS Magnets", 5Th Workshop on Accelerator Magnets in HTS, Budapest 8-12 April 2019.

[6] M. Dam, R. Battiston, W.J. Burger, R. Carpentiero, E. Chesta, R. Luppa, G. de Rijk, L. Rossi, "Conceptual Design of High Temperature Superconducting Magnet for a Particle Physics Experiment in Space", Supercond. Sci. Technol. 33 (2020). doi:10.1088/1361-6668/ab669b

[7] JS. Murtomaki, J. Van Nugteren, G. Kirby, L. Rossi, A. Stenvall, "3-D Mechanical Modeling of 20 T HTS Clover Leaf End Coils – Good Practices and Lessons Learned", IEEE Trans. Appl. Supercond. 29 (2019). doi:10.1109/TASC.2019.2899317

[8] E. Felcini, L. Bottura, J. Van Nugteren, G. de Rijk, G. Kirby, B. Dutoit, "Magnetic Design of a Superconducting Toroidal Gantry for Hadron Therapy", MT 26 conference, 2019. doi:10.1109/TASC.2020.2966174

[9] D. Schoerling, "HTS Program at CERN", Muon Collider Workshop, April 2020, CERN, Switzerland.

[10] F.O. Pincot, "HTS Project: 7 Tapes Winding System", CERN, EDMS 2339258.

[11] CompactRIO Systems, https://www.ni.com/en-us/shop/compactrio.html

[12] S.C. Richter et al. "HTS Undulator Vertical Racetrack (VR) Coil Design", EDMS 2612713, August 2021, CERN.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

CONTINUOUS INTEGRATION FOR PLC-BASED CONTROL SYSTEM DEVELOPMENT

B. Schofield*, J. Borrego, E. Blanco, CERN, Geneva, Switzerland

Abstract

Continuous Integration/Continuous Deployment (CI/CD) is a software engineering methodology which emphasises frequent, small changes committed to a version control system, which are verified by a suite of automatic tests, and which may be deployed to different environments. While CI/CD is well established in software engineering, it is not yet widely used in the development of industrial controls systems. However, the advantages of using CI/CD for such systems are clear. In this paper we describe a complete CI/CD pipeline able to automatically build Siemens `Simatic` Programmable Logic Controller (PLC) projects from source files, download the program to a PLC, and run a sequence of tests which interact with the PLC via both a Simulation Unit Profibus simulator and an OPC Unified Architecture (OPC UA) interface provided by `Simatic NET`. To achieve this, a Google Remote Procedure Call (gRPC) service wrapping the `Simatic` Application Programming Interface (API) was used to provide an interface for interacting with the PLC project from the pipeline. In addition, a Python wrapper was created for the Simulation Unit API, as well as for the OPC UA interface, which allowed the test suite to be implemented in Python. A particle accelerator interlock system based on Siemens S7-300 PLCs has been taken as a use case to demonstrate the concept.

INTRODUCTION

In software engineering, Continuous Integration (CI) refers to a method of development in which changes are regularly incorporated into a central repository. It is very often associated with the presence of build and test automation, in which code is automatically compiled, and a set of tests run. The objective of the methodology is to provide early detection of bugs introduced by code changes, and to simplify workflows in the case where there are several developers working on a single code base (the alternative is to perform periodic merges of the individual developers' branches, which may be very complex if many changes have been made).

Continuous Deployment (CD) is perhaps less well defined, and entails at least the automatic release of some artefact of the automated build process in the CI stage. It may also involve the fully automatic deployment of the artefact in a production environment.

Software for PLC-based control systems traditionally does not follow CI/CD principles, for a number of reasons. Generally, proprietary engineering tools are used as the development environment, in which code is written, compiled and downloaded to the PLC.

* Corresponding Author. E-mail: brad.schofield@cern.ch

In general there is no support for external version control systems to be used for the source code, at least as far as CERN's standard PLC suppliers are concerned (Siemens, Schneider). Instead, often the full project is included in version control as the collection of files and data used by the engineering tools, provided one is using version control at all. Tracking code changes is difficult, and merging branches is effectively impossible with such a workflow. Automation of the building of PLC projects is not trivial, as not all engineering tools provide easy access from scripting languages to these functionalities. Finally, automated testing proves to be cumbersome for a number of reasons that will be elaborated in later sections.

The question addressed in this article is whether it is technically feasible to implement a CI/CD workflow for PLC-based controls development, and if so, whether such a workflow is practicable and useful in a real-world application.

In order to address the first point, set of tools will be introduced which aim to overcome the obstacles preventing the adoption of CI/CD for PLC-based control system development. Tools for automation of the build process are proposed, with the focus on Siemens `Simatic` applications, although tooling for other engineering tools has also been developed. An approach for implementing automated testing of the complete PLC program is described, consisting of an interface to a fieldbus simulator, as well as an OPC UA interface to the PLC and Supervisory Control And Data Acquisition (SCADA) layers of the control system.

To demonstrate these tools, and illustrate their potential, a use case consisting of an interlock system for the Large Hadron Collider (LHC) is presented. Major updates and refactoring of this control system have been enabled by employing the CI/CD workflow presented in this article.

The article begins with addressing the question of the tooling required to automate the building of PLC projects. After that, automatic testing is addressed. Finally, the use case is explained and details of the proposed workflow are given within the context of that project.

TOOLS FOR AUTOMATING PLC PROJECT BUILDING

Intended Workflow

In order to implement CI/CD for PLC-based applications, it is necessary to adopt a similar underlying development workflow to that used elsewhere in software engineering. Fundamentally, this means adopting the use of version control for all source code. The source code is then compiled to produce some form of output, for example executable programs or libraries, for one or more target systems. These outputs in themselves do not need to be included in version

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

control, as they can simply be recreated from a specific state of the versioned source code. The outputs are often referred to as *artefacts*. The CI/CD approach is to automate this process of creation (and subsequent deployment) of artefacts.

In the majority of mainstream programming languages, the source code simply consists of text files. The most widely used version control systems are therefore also based on the use of text files. If it is intended for the complete source of a PLC program to be stored in a version control system, this introduces some restrictions. Of the IEC-61131-3 languages, only the languages which can be represented in text, namely Structured Text (ST) and Instruction List (IL) are suited to this approach. However, the requirement to use non text-based PLC languages does not preclude the use of the proposed approach; it may be the case that a part of the code base is implemented in text-based languages, and other parts in non text-based languages, of which the former are tracked in an external version control tool and the latter kept in the PLC engineering tool.

PLC Engineering Tools

As previously mentioned, Programmable Logic Controller (PLC) engineering must be performed using the proprietary engineering tools supplied by PLC vendors. The interface for such tools is typically a Graphical User Interface (GUI), in which importation of code from external sources, code editing, compilation, configuration of the target, and downloading of the program are performed. In general, these tools do not have good support for version control systems, meaning that it is difficult to maintain consistency between source code that is maintained in version control, and the resulting compiled version of a project. This is in essence the core problem being treated in this paper. While it is possible to use dedicated version control tools for PLC projects such as `versiondog` [1], they lack some of the more advanced features of version control systems employed in standard software development like `git`. They also implicitly track all changes to the compiled project, whereas the desired workflow outlined here aims at keeping only the text-based sources in version control, excluding any derived binaries and other non-essential data files. In order to achieve the desired workflow, it is necessary to automate tasks ranging from importation of external sources, to compilation and ultimately downloading of the project to a PLC. If this can be achieved, then the CI/CD workflow will be able to guarantee consistency between source files in the repository and a program running on a PLC, something which is currently not possible with existing tools.

In this work we focus on the engineering tools for Siemens S7-300 PLCs, since these are present in the system at hand. These PLC projects are developed using `Simatic Step 7`, which exposes a C#/Visual Basic API. We developed a C# library which allows us to import source files, build the project and download it to either a test or production PLC. We then designed a simple gRPC service, allowing our C# server to fulfil requests and interact directly with the Step7 library, which can be launched from remote clients, implemented in whichever language is convenient. We use a Python client due to its simplicity of use and to match the environment for running the integration test suite. Figure 1 shows the client/server architecture for the proposed S7 gRPC service.

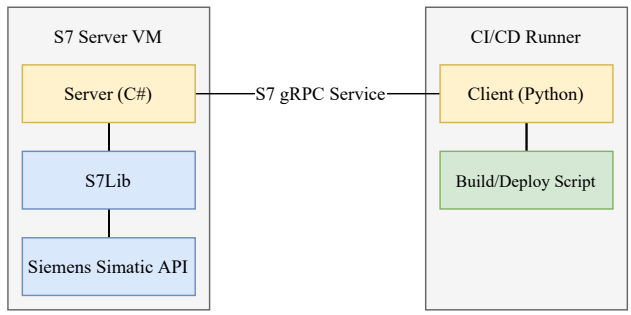


Figure 1: Architecture of the `Simatic` build tools.

The S7 gRPC service exposes several methods, namely:

- `CreateProject` to create an empty `Simatic` project given the project name and destination path;
- `ImportSource` to import a source file into project;
- `ImportSymbols` to import a symbol table file into project;
- `CompileSources` to build target sources into blocks;
- `CompileAllStations` to build the hardware configuration for each station in the `Simatic` project;
- `StartProgram` to start/restart a specific program on the target CPU;
- `StopsProgram` to stop a specific program on the target CPU;
- `DownloadProgramBlocks` to download a specific program to target CPU.

Finally, we have also developed similar tooling to support newer Siemens S7-1500 and Schneider CPUs, although these are not in the scope of this article.

AUTOMATED TESTING FOR PLC PROGRAMS

With automated build and deployment tools in place, the last remaining piece required to implement a complete CI/CD workflow is the automated test suite. Automated testing of PLC programs is challenging for several reasons. For the majority of modern programming languages, there are numerous unit testing frameworks available, which make the implementation of unit tests almost trivially simple. In addition, it is generally not difficult to create a build configuration which will create test executables which can be run locally, for example on the developer's machine. This is

[1] `https://auvesy.com/en/versiondog`, as of October 21, 2021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

however not the case for PLC programming languages. In order to implement true unit tests for PLC code, frameworks specific to a PLC type would have to be implemented. This topic is explored in [1], in which a unit testing framework for Siemens PLCs is developed. For integration testing, the problem is further complicated by the difficulty of 'mocking' the necessary interfaces (for example fieldbuses and supervision systems). Regarding the target on which the test code shall be run, in the case of PLCs this must be either a physical PLC CPU, or a simulator. While the simulator approach may be attractive, currently not all PLC simulators support the functionality required to implement complete test suites.

The approach to PLC program testing proposed here is to generate a PLC program which is either identical, or very close to the final program required in production. This program is then deployed to a physical PLC. The test suite is then implemented externally, and utilises a number of interfaces to interact with the PLC in order to run the tests. The interfaces to the PLC can be summarised as acting on three levels, namely the input-output or fieldbus level, the PLC program level, and the supervision level. These interfaces will be described in detail in the next sections, before describing how they can be combined into a single test suite.

In software development there's a clear distinction between unit testing, integration testing and system testing. Unit tests focus on verifying the correctness of individual functions or modules in isolation. Integration tests aim at ensuring the module's interface is correct and it behaves as expected when interacting with other modules. Finally, system testing verifies that a completely integrated system fulfils its requirements. Our proposal cannot be said to consist of unit tests, at least as far as the PLC source code is concerned. Instead we introduce a set of tests which verify that the control system fulfils its task accordingly, which is more strongly aligned with system testing. Even though we use several communication interfaces and thereby implicitly test them, they are not the focus of the test, so they cannot be considered integration tests. Nevertheless, we still mock the PLC inputs and evaluate outputs at both the PLC and SCADA layers.

Fieldbus Simulators

A PLC application will almost invariably need to interact with the physical world via sensors and actuators. The input-output (IO) hardware required to do this may be installed locally to the PLC CPU, but is more generally installed in a distributed manner, and connected with the PLC via a fieldbus. The fact that this interface is standardised can be exploited for testing purposes, if simulators for such fieldbuses are available. In this article, the focus is on Siemens PLCs, and therefore Profibus and Profinet fieldbus simulators will be taken as examples. Siemens provides hardware which allows the simulation of arbitrary hardware configurations for both Profinet and Profibus. The units are controlled via an Ethernet interface, and a C API is available. For the purposes of automatic testing, such a simulator can be connected to a test PLC, and a CI/CD pipeline could be used to configure the simulator (by loading a specific hardware configuration), and control IO values, both via the API. In order to more easily access the functionality of the simulation unit from the automated build and test suite, a Python wrapper `py-simulation-unit` was created.

PLC Interaction via OPC UA

While the fieldbus simulator provides a way to mock field level input to the PLC, from the point of view of writing tests it may be valuable to access the internal state of the PLC program. This is fairly straightforward to achieve provided the target PLC can expose tagged variables in an OPC UA server. In the case of Siemens PLC, Siemens `Simatic NET` software provides such an OPC UA server, which runs in a separate workstation. Newer generations of PLCs also have onboard OPC UA servers, further simplifying the setup.

In [2], OPC UA was used as the exclusive interface for testing PLC applications. This was possible because the applications under test were all created with the UNICOS framework [3], the structure of which provides a simple way to 'override' the hardware IO interface with values written over OPC UA. This effectively means you can mock the IO at the PLC program level. Depending on the way in which an application is written, it may not always be possible to do this, and IO simulation may be necessary.

In this work, the OPC UA client API from the open source LGPL Pure Python OPC UA[2] package is used.

Supervision Systems

A PLC-based control system most often interacts with a SCADA system, in which operators can observe the status of the system, and send commands to the PLC. In certain cases there may be additional logic implemented in the SCADA system, which provide further inputs to the PLC. Rather than mocking the SCADA interface (which in the use case presented in this article consists of Siemens S7 communication with a CERN-specific protocol built on it), it was decided to instantiate a SCADA system specifically for the test setup, and to interact with it via OPC Unified Architecture (OPC UA). This interaction allows the test suite to 'force' operator actions and logic outputs to the PLC, and thus provides a very realistic test setup. An added benefit is that the test suite can be used for manual validation of the graphical features of the SCADA system, as shown in Fig. 2. Effectively, by visual inspection of the SCADA faceplate it is possible to monitor the automated test execution in the lab setup. The SCADA system used here is built with WinCC OA, using features from CERN's UNICOS framework [3].

[2] `https://github.com/FreeOpcUa/python-opcua`, as of October 21, 2021

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

Figure 2: Part of the SCADA faceplate for one Powering Interlock Controller PLC. The automated test suite can be used to drive inputs to manually validate the supervision layer.

USE CASE: LHC POWERING INTERLOCK CONTROLLER

An example of the utilisation of the proposed CI/CD workflow can be found in the Powering Interlock Controller (PIC) of the LHC. This system controls the powering of the superconducting magnets of the LHC, and consists of 36 Siemens S7-319 PLCs, physically distributed around the 27 km circumference of the LHC as shown in Fig. 3. Each PLC is responsible for managing the interlocks for a certain number of electrical circuits powering the magnets. Depending on the types of magnets being controlled, the PIC is required to monitor different types of signals, and perform different interlock actions. All 36 PLCs utilize a common generic code base, which is parameterized in each PLC using source files which determine input-output mappings and circuit types [4].

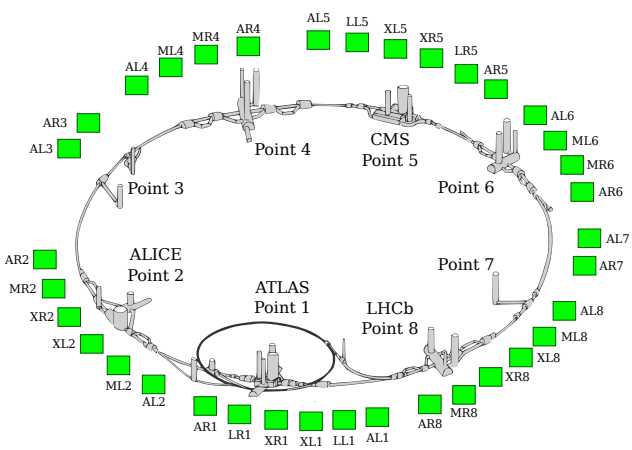


Figure 3: Layout of the LHC showing PIC PLC locations. Each PLC is shown as a green box with the corresponding name.

Legacy Workflow

Prior to implementing a CI/CD workflow, the method used to update the generic code was to edit it in each `Simatic` project. This was time-consuming and error prone. Deployed versions of the PLC projects were kept in `versiondog`, which enabled tracking of changes to each individual `Simatic` project. Testing was performed manually with a hardware-based lab setup. Consistency between the code depoloyed on the lab setup, and that deployed in the production system had to be handled by the developer.

CI/CD Workflow

The starting point of the CI/CD workflow was the creation of a `GitLab` repository containing a set of sources (ST and IL files) for both the generic code (the same for all PLCs) and the configuration code (different files for each PLC). A set of 'base' `Simatic` projects, containing the hardware configurations of each PLC, are also included in the repository. It should be noted that these projects are only used as inputs to the workflow; that is, the resulting complete `Simatic` projects are not versioned in `git`.

Lab testing workflow The automated testing is performed in the lab, using an identical PLC to those used in production, connected to a Siemens Profibus Simulation Unit. An additional `Simatic` project containing the configuration of this PLC is present in the repository; the hardware configuration of which is the basis of the Simulation Unit configuration.

The build stage of the CI/CD workflow uses a Python build script to import the generic sources as well as the specific configuration sources of the configuration to be tested. The resulting program is compiled. Finally, the complete PLC program is downloaded to the lab PLC. In order to update the OPC UA server used for the testing, the script also downloads to the OPC UA station.

With the build stage complete, the test stage can begin. This simply involves calling `pytest` on the complete test suite. Test fixtures manage the connections to the Simulation Unit, as well as the OPC UA servers for both the PLC and the SCADA. An example of a `pytest` fixture is as follows (for the PLC client):

```
@pytest.fixture
def plc_client() -> Client:
    """Returns an OPC UA client connected to the PLC"""
    client = Client(CLIENT_PLC_URL)
    client.connect()
    yield client
    client.disconnect()
```

Listing 1: Fixture for OPC UA PLC Client

The SCADA client fixture follows the same pattern. The simulation unit fixture (called `simba` in this use case) is slightly more complex as the hardware configuration is loaded.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A specific test may use any combination of these fixtures, depending on what needs to be accessed in the test. The following is a very simple test to verify the magnet quench detection status for the main dipole magnet circuit:

```
def test_quench_abort_status_rb(plc_client, simba):
    """Main Dipole Quench Status test"""
    plc = plc_client.get_root()
    status_abort = plc.get_child("7:A_A_1_ST_ABORT")
    quench_status =
     ↪  plc.get_child(['7:1_Instance1_Circuit1',
     ↪  '7:Quench_Status'])

    simba.write_io_bin('S000I4.1', 1)
    wait()
    assert status_abort.get_value() is True
    assert quench_status.get_value() is False

    simba.write_io_bin('S000I4.1', 0)
    wait()
    assert status_abort.get_value() is False
    assert quench_status.get_value() is True
```

Listing 2: Example Test for the Power Interlock Controller System

It can be seen that the inputs are manipulated using their hardware addresses directly, via the `simba` fixture. If desired, an abstraction layer based on the signal list of a specific project could be implemented to simplify the reading and writing of the tests. The test assertion is based on accessing the instance data block of a specific function block representing a state machine corresponding to the main dipole circuit, and verifying that a particular property is correctly set. This illustrates the usefulness of the OPC UA connection to the PLC, as this information may not necessarily be accessible via other interfaces such as the SCADA.

Naturally, the more complete functionality tests are more complex than the examples reproduced here, but all tests follow the same principles. All three interfaces are used to drive the program to the desired initial state, and then test inputs can be given, representing either field or operator input. The interface to the internal PLC data allows many 'fine-grained' assertions to be made, such that if a test fails, it is easy to identify the location in the code that caused the failure.

LHC build workflow With the tests passing, it is now possible to build all 36 PLC projects ready for deployment to the LHC. For this build stage, the build script is parameterized by the different subsectors of the LHC, allowing some control over which PLC projects are built. This can be useful as interventions are typically performed on one sector of the LHC at a time, and the complete build pipeline for all 36 projects takes approximately 2 hours to run. Currently, automatic deployment is not performed for the production PLCs although there is no technical barrier for this. Instead, the artefacts of the build pipeline (i.e. the `Simatic` projects) are added as a new version to `versiondog` and downloaded manually. The complete conceptual workflow is illustrated in Fig. 4.

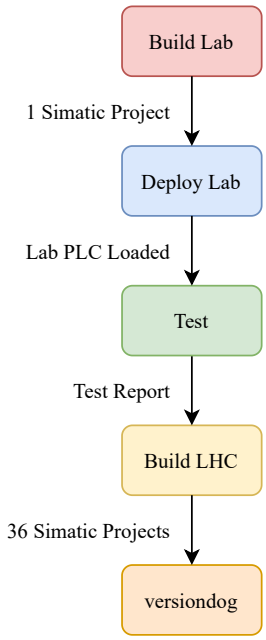


Figure 4: The conceptual CI/CD workflow. Labels by the arrows describe the artefacts produced by the preceding stage.

Tangible benefits of the CI/CD workflow In terms of the time saved for simply deploying a code change, the current workflow (with manual download) is estimated to reduce workload by approximately an order of magnitude. Manually updating each project would take at least a day, whereas simply downloading the resultant artefacts can be done in approximately one hour. With automated deployment, no developer input would be required after committing the changes to the repository and triggering the pipeline.

Naturally, the development time saved by having access to a full automated test suite is not as easily estimated, and must also be offset by the time taken to develop and maintain the test suite itself as well as the Continuous Integration/Continuous Deployment (CI/CD) infrastructure. Nevertheless, in the scope of this use case it has been clear that the CI/CD workflow has greatly facilitated development of new features, the refactoring of the code, as well as identification and resolution of bugs.

CONCLUSION

In this article a complete CI/CD workflow for PLC application development is proposed, along with a set of tools which enable all of the necessary stages of such a workflow. In summary, these tools are:

1. A gRPC service enabling scripting of many operations needed during PLC project engineering, including importation of sources, compilation, and download;

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV035

2. A Python wrapper for the Siemens Simulation Unit API, allowing configuration and manipulation of simulated IO from Python;

3. An example of an automated test suite, implemented in Python and taking advantage of existing testing packages (`pytest`) as well as an open source OPC UA package to interface with the PLC and SCADA.

The complete CI/CD workflow has been illustrated using the Powering Interlock Control system of the LHC.

Future Work

Currently, our test suite still refers to each simulated Profibus I/O channel by a tag name defined in the Simulation Unit project. A simple yet effective improvement would be to create an additional shallow abstraction layer that maps a meaningful device field to each I/O tag name. This would considerably improve the test readability.

Additionally, it seems feasible to automatically upload the PLC project built by the CI pipeline directly to versiondog, preserving a connection to a tagged commit in version control. This would effectively give us the best of both worlds: being able to quickly deploy to a running PLC while knowing exactly which source code is responsible for that specific version of the program.

Our current proposal still relies on manually creating base projects which include the hardware description. The need for these could be eliminated by just creating an empty project, automatically generating the hardware configuration and importing it as part of the CI/CD workflow.

Finally, one can think of how we can effectively implement unit testing at the PLC source code level by leveraging PLC simulators, namely with support for cycle-by-cycle execution, as is explored in [1].

REFERENCES

[1] G. Sallai, E. B. Viñuela, and D. Darvas, "Testing Solutions for Siemens PLCs Programs Based on PLCSIM Advanced," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1107–1110, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-WEPHA018. https://jacow.org/icalepcs2019/papers/wepha018.pdf

[2] B. Schofield, E. B. Viñuela, and J. Borrego, "Continuous Integration for PLC-based Control Systems," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1527–1531, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-WESH4003. https://jacow.org/icalepcs2019/papers/wesh4003.pdf

[3] P. Gayet, R. Barillere, *et al.*, "UNICOS a framework to build industry-like control systems, Principles and Methodology," in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'05)*, Geneva, Switzerland, 2005.

[4] M. Zerlauth, C. Dehavay, B. Puccio, R. Schmidt, and E. Veyrunes, "From the LHC reference database to the powering interlock system," in *9th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2003)*, Oct. 2003, pp. 395–397.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV036

AN EVALUATION OF SCHNEIDER M580 HSBY PLC REDUNDANCY IN THE R744 SYSTEM A COOLING UNIT

D. Teixeira†, University of Cape Town, Cape Town, South Africa
L. Zwalinski, L. Davoine, W. Hulek, CERN European Organization for Nuclear Research, Geneva, Switzerland

Abstract

The Detector Technologies group at CERN has developed a 2-stage transcritical R744 cooling system as a service for future detector cooling. This is the first system in operation at CERN where Schneider HSBY (Hot Standby) redundant PLCs are used. This cooling system provides a good opportunity to test the Schneider redundant PLC system and understand the operation, limitations and probability of failure in a controlled environment. The PLC redundancy is achieved by connecting Schneider M580 HSBY redundant PLCs to the system where one is the primary which operates the system and the other is in standby mode. A series of tests have been developed to understand the operation and failure modes of the PLCs by simulating different primary PLC failures and observing whether the standby PLC can seamlessly take over the system operation.

INTRODUCTION

Previously, most large-scale systems at CERN have made use of multiple small PLCs where there is one running per subsystem and communicating with one leading PLC. The idea behind this is that if one PLC fails, the other subsystems are still able to operate. However, in previous experiences, losing one subsystem can cause erroneous readings fed to the other subsystems or interlocks created in the processes of the other subsystems.

In the context to avoid multiple semi-autonomous PLCs, EP-DT decided to go in the direction of a central powerful PLC connected to the different subsystems which each have communication cards that read and write inputs and outputs. This ensures the centralisation of information inside a single core element. In order to improve the communication availability of such systems the RIO loop was proposed following ring topology. This has been used in the MAUVE system with good results. The ring topology used is shown in Fig 1.

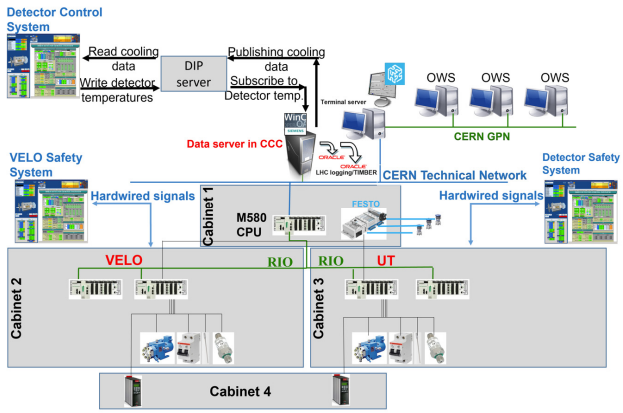


Figure 1: MAUVE control architecture.

For the LHC Phase II upgrade, two different cooling systems, CO_2 based, will be installed, namely the R744 Primary system (2 stages transcritical) [1] and the 2PACL (2 Phase Accumulator Control Loop) system [2], both of which are quite large with a high quantity of inputs and outputs.

The approach of a main core system to control the different remote IO was foreseen to be used in Phase 2. However, following recent technological developments by Schneider, the solution of PLC redundancy was proposed to improve the reliability of the system. Before installing the systems underground, this solution was first implemented on the surface with R744 Primary System A which is currently operational. Following this, it will also be used in the DEMO system which is a 2PACL equivalent currently being installed. The purpose of these surface systems is to gain experience on the process and validate the redundant architecture before installing the final systems underground.

CONTROLS

PLC Selection Following Current Solutions Available on the Market

The Schneider M580 HSBY redundant PLC was chosen for these systems due to satisfactory compatibility with the currently used UNICOS (Unified Industrial Control System [3]) framework as well as the simplicity of implementation. CERN makes use of mostly Schneider and Siemens PLCs, with Schneider being the most commonly used PLC in cooling applications. PLC redundancy has previously been implemented at CERN using Siemens PLCs (S7-400). However, the return of experience from our cryogenics colleague experts was giving a global difficulty with implementation and possibility to update code. There is a new type of PLC from Siemens which proposes a redundant

† TXRDAN001@myuct.ac.za

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV036

functionality. However, these PLCs (1500 generation) use dynamic addressing which is currently not the simplest to implement with UNICOS framework as this last one is establishing a link between PLC and SCADA through fixed addressing. This leads to a need of global update on both SCADA and PLC for any simple operation. The Schneider redundant PLCs used in the new systems are much simpler to maintain and program using the CERN environment.

From the Electrical System Configuration to the Control System

Following the idea of improving the reliability and availability of the cooling systems, several decisions have been made in terms of electrical distribution and control systems. After performing an analysis of critical equipment, the most important elements are connected to a UPS (uninterrupted power supply) to avoid system failure during loss of power. To ensure circulation of CO_2 for example, the liquid pump creating the flow in the system is under UPS.

It is also critical to keep the 24VDC circuit operational. The approach here is to use two redundant power supplies, interconnected through a redundancy module, with one power supply under UPS. The configuration of the system can be seen in Fig. 2 below.

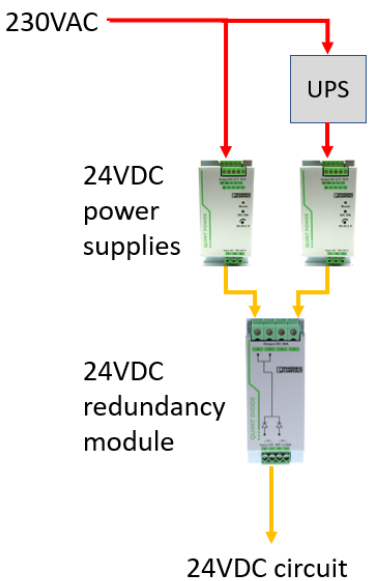


Figure 2: 24VDC Redundant power supply configuration.

The benefit of this configuration is that the failure of one power supply does not cause a loss of power to the 24VDC components and signals, meaning that we keep the monitoring possibility of the system and can monitor and track the current process even in case of interlock. This is needed for system diagnostic to understand the process state at any time.

Similarly, the use of redundant PLCs aims improve availability by removing the single point of failure found in single PLC applications. The system has two backplanes with two separated PLCs. Each PLC has its own communication card which communicate with the CERN Technical Network. Risk of failure is decreased by powering each PLC with its own dedicated redundant 24VDC power supplies. The configuration of the redundant PLCs used in the R744 Primary System is shown in Fig 3.

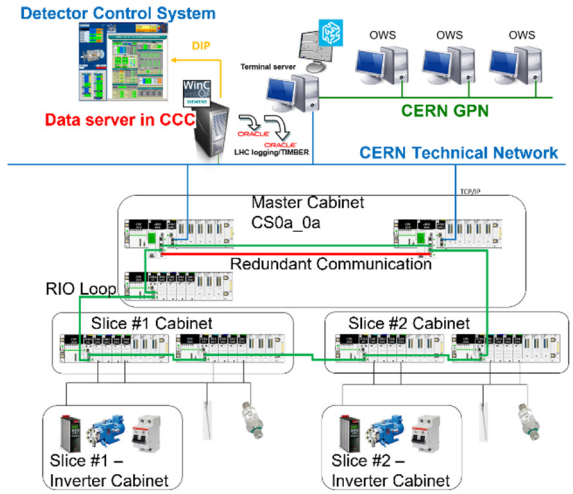


Figure 3: Topology of PLCs in R744 Primary System.

The green connections represent the RIO loop which exchanges data between backplanes. The red connection between the two PLCs represents the Hot Standby link which sends application variables, system status and I/O data from the primary PLC to the standby PLC. It is also used to compare the code program stored in each PLC [4].

The PLCs are named PLC A and PLC B, either one receives the role of primary PLC and standby PLC. This depends on several parameters such as which PLC was loaded first, internal logic comparison, and the state of the PLCs.

Both PLCs run the same program, with one acting as the primary PLC which operates the system and the other acting as the standby, ready to take over should the primary PLC become unavailable.

Additionally, efforts have been made to reduce errors in the system by using automatic code generation [5]. There are many basic and complex functions that are needed several times in each project, such as addition, subtraction, and saturation temperature calculations. Templates are written with the logic for each basic function and are then used to generate multiple similar functions where the PLC developer needs only to specify the names or values of the input variables for each instance. This way, the function is written once and can be used to generate as many iterations of the function as required. This reduces the likelihood of syntax errors which are common when developing code.

Failure Modes

The Schneider HSBY redundant PLCs have already been installed in the R744 Primary System A and this system was used to test the various failure modes of the PLC, power supply, and communication with the technical network and remote IO backplanes. It is important to understand how the system operates under different combinations of failure modes and whether the system can still run under these circumstances. The results of these tests are summarised in Table 1. In these tests, a PLC failure is

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV036

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

defined as any loss of PLC operation, whether due to a failure on the power supply, a circuit breaker tripped, or a failure within the PLC itself.

Table 1: PLC and System Response to Multiple Failure Modes

Failure mode	PLC A	PLC B	System Response
1. PLC B failure	Primary	Off	Running
2. PLC A failure	Off	Primary	Running
3. Failure of both PLCs	Off	Off	Not running
4. PLC A loss of network communication	Standby	Primary	Running
5. PLC B failure & PLC A loss of network communication	Primary	Off	Running, no network communication
6. PLC B loss of network communication	Primary	Standby	Running
7. PLC A failure & PLC B loss of network communication	Off	Primary	Running, no network communication

In addition to the failure modes shown in Table 1, there are failure modes which result in more complex PLC and system behaviour. These cases are described in detail in Table 2.

Table 2: PLC and System Response to Complex Failure Modes

Failure mode	PLC Behaviour and System Response
8. Logic of PLC A different to PLC B	The PLC which is initially primary remains as such and the other PLC is in wait mode. The system operates without error.
9. Loss of one link between backplanes	The primary and standby PLCs remain as they were, and the system operates as normal
10. Loss of multiple links between backplanes	The primary and standby PLCs remain as they were, but the system runs with loss of functionality. The backplanes which are completely cut off from the PLCs are inoperable.
11. One PLC is completely disconnected from the RIO loop, the other has no connection to the Technical Network, the Hot Standby link is still available	The PLC connected to the Technical Network becomes the primary and the PLC which is still connected to the RIO loop is in standby. In this case the system does not operate as the primary PLC has no communication with the remote backplanes

From the system responses seen in Tables 1 and 2, it is clear that the only failure modes that result in loss of system functionality or full system failure are when both PLCs fail or multiple connections in the RIO loop are lost.

In addition to reducing the likelihood of failure, changing the primary PLC from PLC A to PLC B happens seamlessly with no effect seen in the system.

The switching between primary and standby modes makes use of the default switching logic from UNICOS, however, the developer does have the freedom to define user-specific conditions for this change. For example, the R744 Primary System A will make use of user-defined logic to modify the scenario shown in point 11 of Table 2. In the case that one PLC is disconnected from the RIO loop and the other is disconnected from the technical network, the default logic would keep the PLC connected to the Technical Network as the primary despite its lack of communication with the other remote backplanes. This is not consistent with the reliability definition established for the Phase 2 upgrade. Instead, the system should have the primary PLC connected to the RIO loop while the PLC connected to the Technical Network should be in standby mode. In this case, there is no communication of system information to the network, but the system can still operate.

CONCLUSION

The Schneider HSBY redundant PLCs are a solution compatible with the Phase 2 cooling upgrade requirements in term of availability and reliability. Its implementation is supported by CERN UNICOS framework and specific automated generation for CO_2 templates developed by EP-DT. The introduction of PLC redundancy is defined as baseline for the LHC detectors' critical CO_2 cooling applications as an additional level of safety.

REFERENCES

[1] Barroca P. *et al.*, "An Ultra-Low Temperature Transcritical R744 Refrigeration System for Future Detectors at CERN LHC", Applied Sciences, Aug. 2021, 11, 7399. https://doi.org/ 10.3390/app11167399

[2] Verlaat B. *et al.*, "CO2 cooling for the LHCb-VELO experiment at CERN", 8th IIF/IIR Gustav Lorentzen Conference on Natural Working Fluids, Copenhagen, Denmark, CDP 16-T3-08.

[3] Gayet Ph. *et al.*, "UNICOS a Framework to Build Industry Like Control Systems: Principles & Methodology", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper WE2.2-6I.

[4] "Modicon M580 Hot Standby: System Planning Guide for Frequently Used Architectures," *Schneider Electric*, Sept. 2020, Available: https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=NHA58880.08.pdf&p_Doc_Ref=NHA58880

[5] Zwalinski L. *et al.*, "First Steps in Automated Software Development Approach for LHC Phase II Upgrades CO_2 Detector Cooling Systems", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, paper WEPHA170, pp.1488-1491. doi:10.18429/JACoW-ICALEPCS2019-WEPHA170

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV037

MODULAR SOFTWARE ARCHITECTURE FOR THE NEW CERN INJECTOR WIRE-SCANNERS

A. Guerrero†, D. Belohrad, J. Emery, S. Jackson, F. Roncarolo,
European Organization for Nuclear Research (CERN), Geneva, Switzerland

Abstract

In the scope of the LHC injector upgrade, new wire-scanner devices have been installed in the LHC injector circular accelerators. This paper outlines the software architecture and choices taken in order to provide the scanner experts with comprehensive diagnostics as well as operators with straightforward size measurements. The underlying electronics acquire large amounts of data that need to be accessible for expert and machine development use and need to be processed before being presented for daily operational use, in the shape of a beam profile and its derived size. Data delivery and measurement computation are accomplished by means of a modular structure, using functionally distributed real-time processes that handle the different data views, with minimal interference in the processing, and minimal exchange of data among modules.

INTRODUCTION

A wire-scanner is generally based on a thin wire traversing at high-speed circulating particle beams. Monitoring turn by turn the distribution of secondary particles generated by the beam-wire interaction (bottom plot in Figure 1) as a function of the wire position (top plot in Figure 1), allows reconstructing the transverse beam profile.

Figure 1: The top plot shows position data for both in and out scan. On the bottom plot beam profile data is displayed.

Such measurements are performed daily by accelerator operators and experts to infer the size of the beam and via the beam optics and beam momentum spread the transverse emittance of the different beams played in the machines.

At CERN a new generation of Beam Wire-Scanners (BWS) has been developed, installed, and recently commissioned [1] in the scope of the LHC injector upgrade (LIU), during the accelerator restart after the long shutdown (LS2). New engineering concepts were applied to build the hardware and electronics of these devices [2] [3] [4] [5], thus giving rise to a new complete software system built and constrained within the available controls infrastructure and underlying standards. In particular, the Front-End Software Architecture (FESA) [6] [7] was chosen as the framework for the design and development of the system modules.

SYSTEM DESCRIPTION

The BWS applies key innovations to its kinematic unit: moving parts located only in vacuum, magnetic and optical techniques to power and air vacuum signal exchange. The unit performs trajectory control of the shaft via a solid rotor resolver [8] and provides carbon wire measurement and a high accuracy position measurement using an optical encoder [9]. Electronics driving the system consist of an Intelligent Drive (BWSIDC) in charge of the powering and wire movement and an acquisition crate collecting data from the different actuation parts as well as from the optical encoder measurements. Diagnostics and tests possible in *local mode* at this level include verification of cabling and parts of the kinematic subsystem, powering and scanning procedures without beam interaction for diagnostics, all without tunnel access.

Communication with kinematic control and acquisition electronics is done through Ethernet by means of the IPbus protocol and associated module and libraries for firmware and software development [10]. The IPbus protocol is a simple packet-based control protocol for reading and modifying memory-mapped resources within FPGA-based IP-aware hardware devices which have a virtual A32/D32 bus, thus enabling the replacement of VME control (a usual standard in our operational instrumentation) in our case.

The secondary particle shower detection is performed by a scintillator coupled to four Photo-Multiplier Tubes (PMT) to acquire four signal amplitude ranges with the aim of covering the large dynamic range of beam energies and intensities across the LHC injectors [11]. The PMT devices are powered with a custom board optimised for large pulse mode and fast recharge. The gain of the PMT is set via a high voltage power supply controlled through a 4-channel commercial board accessed through the VME bus. PMT output currents are amplified in two stages and driven into parallel high-speed digitizers, feeding the so called VFC [12]. The VFC is a CERN FPGA-based multipurpose carrier with VME interface, designed to be the new standard acquisition platform for the Beam Instrumentation Group. The VFC wire-scanner application firmware was built in house and adapts to the different accelerator synchronisation. The acquisition takes place asynchronously to the bunched beam at 500MHz and with a 14-bit resolution. Together with the sensor current outputs, the beam revolution frequency and bunch timing are written into a memory bank composed of two 8Gb DDR chips. On request, acquired digitised beam information is integrated on the fly and placed in the VME bus for the CPU to recuperate

† ana.guerrero@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

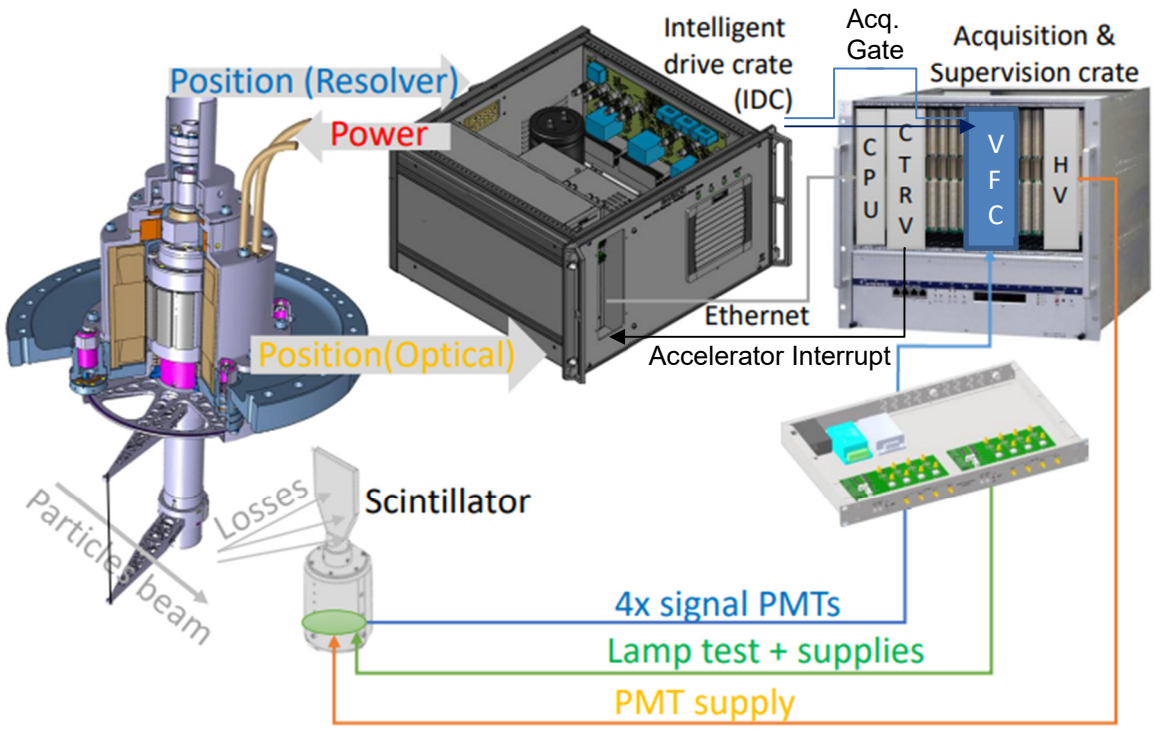


Figure 2: From left to right, mechanics installed in the accelerator, secondary particle detector, IDC, digitisers and VME crate containing the different VME acquisition and control boards.

through VME fast block data transfer. Data transfer is performed using an in-house product called EDGE that automatically generates a driver from a database description. The VFC firmware also controls the amplifier stage and a lamp for test purposes through an UART interface.

The synchronisation to machine timings of the different elements involved in the measurement following a user's request for a specific time in a concrete cycle is achieved by programming an interrupt in a VME timing receiver board (CERN product) that will send a pulse to the BWSIDC. The controller is then triggering the start and end of the acquisition gate in between rotational shaft angles corresponding to the window of measurement provided by the user. These angles can be pre-set using the data derived from a calibration function computed in the laboratory during the pre-installation phase.

Figure 2 shows a diagram of the full hardware system.

The front-end software is running on a MEN-A25 CPU with CENTOS Linux which communicates with the control and acquisition boards either through the VME bus or through Ethernet. It is composed of 5 different FESA classes, each one mostly mimicking the functionality emerging from the hardware design and encapsulating the low-level data corresponding to the main function performed: wire movement across the beam pipe, secondary particle integration for bunch profile construction, PMT working point parametrisation, measurement synchronization and finally operational beam profile measurement.

FESA

Currently, the standard infrastructure for equipment front-end software in the CERN accelerators, the Front-End Software Architecture (FESA) is a complete environment for equipment specialists to design, develop, deploy, and test their equipment software, which produces as an end product a FESA class and FESA devices. The focus of the instrument developer is set on the structure and the flow control of the application domain, and to standardise the outcome of this analysis into a framework. The framework orchestrates the activity, real-time processes and user interface exchanges calling routines provided by the application developer to apply the equipment-specific behaviour.

The internal real-time functionality can be as modularised and parallelised as allowed by the constraints imposed by the underlying hardware components of the system. The outcome of the different threads or concurrency layers talking to the external hardware is stored in a shared memory accessed also by the defined interface. An operational FESA class receives input parameters from database declared fields for server initialisation and from external sources via a predefined interface using a device-property model. The acquired and processed data is retrieved via the same mechanism.

Thus, all data exchanges are done through one server dealing with all the requests coming from the different user scripts and applications accessing the class. In our case, expected applications accessing the class are hardware expert's Python scripts to retrieve hardware status information and raw data, Java and Python expert applications

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV037

to control and display low level information for instrument diagnostics, operational applications to produce measurements and set parameters interesting to operators and accelerator specialists, multipurpose WEB interfaces [13], ad-hoc scripts launched during machine development periods while studying the machine behaviour or the instrument behaviour in test conditions and last but not least, data retrieval for long term storage.

FUNCTIONAL MODULES

As we have seen in the system description, the hardware and electronics composing the BWS system have two fundamental functional modules controlling and acquiring both sensors and actuators, one handling the movement of the wire and one in charge of the particle distribution acquisition. The only link between them is the hardware signal triggering the acquisition gate. The parameters involved in the generation of this trigger are an essential concern for meaningful profile measurements in cycling machines where the cycle time at which the wire is flying in the beam is precisely requested by the user performing the scan and necessarily synchronised to the played cycling sequence. It is this synchronisation functionality using the timing infrastructure of the accelerator which makes up a third functional module and drives the former two modules. These three quasi-independent modules become integrated with one another by means of a fourth module, the operational bunch profile measurement, dedicated to the construction of the requested measurement. For practical reasons, we decided to split the particle distribution acquisition into two different modules, namely bunch profile acquisition and PMT gain thus making up a total of five functional modules.

More and more specialists and operators want to access system data and beam data at different processing levels. The functionality and type of the exchanged data between the FESA server and the users may be as varied as the type of user looking at and/or actuating them, but we can differentiate three main types: machine operators, accelerator specialists and system specialists. The described functional decomposition of the software enables at the same time an easy encapsulation of the data to provide different data views of the system and different security access depending on the user.

Wire Displacement

Hardware-wise the wire displacement is the most complex functionality of the system, however this is not the case at the software level. This module is in charge of starting a scan with or without external synchronisation and of reading a bulk of registers with acquired data and status. No processing of data is required for publishing. The decision to design this module separately also obeys two particularities at the functional internal level, the introduction of the IPBus protocol for the data transfer on one hand and the handling of critical settings on the other. Whereas in the other modules errors in the parametrisation of the devices do not have consequences other than a wrong measurement, this module includes a few parameters that could potentially disturb the feedback in the movement or the lifetime of some of the instrument components. The firmware implements all needed protections to avoid any harm to the instrument but nevertheless a strict policy must be applied to the initialisation and modification of critical settings. The access to the module should be confined to the instrument experts.

Regarding the choice of data transfer between the BWSIDC and the software module, the use of a VFC was initially considered to profit from the tools and automatic driver generation from the EDGE register description. Finally, we opted to use ethernet with IPbus as the communication technology to reduce the VME bus data traffic at the same time as allowing parallelising the data transfer of the modules retrieving most of the acquired raw data. In order to maintain the standard register description, the module initialisation required the incorporation of a translation of the current EDGE description into the XML description supported by the UHAL library of IPbus.

Bunch Profile Acquisition

This module deals with the different aspects of the bunch transverse profile construction from the digitised particle shower distribution detected: It prepares the VFC and amplifier stage for acquisition and reads four integrated data streams and the raw data of a non-saturated channel for publication. From the firmware signal level detection facility, it tries to derive the ideally ranged channel. When requested, attempts to compute the bunch phase with respect to the retrieved or produced bunch clock depending on the accelerator concerned for correct integration of the bunch. Both the PS and PSB injector electronics do not have access to the bunch timing but only to the revolution frequency, however the harmonic used is known before-hand. The bunch timing is thus generated by the firmware dividing the turn length by the harmonic. The phase will depend on the type of beam, scan cycle-time requested and radio frequency signal offset.

The main internal requirement for handling the bunch acquisition is high speed data transfer due to the large amount of raw data produced in the four digitalisation channels.

For test purposes a lamp control is also included in its functionality.

PMT Gain

One could argue that this functionality can be integrated in the bunch profile acquisition module since it is an essential working component of the beam signal amplitude. However, it is distinctly decoupled from the hardware point of view and at the same time, needs no dynamic control which is a key point for adding this module. As the system has been conceived, the working point of the PMT filter combination needs tuning only during a commissioning phase to correctly cover the ranges of energy and intensity without neither saturating all channels nor decreasing the signal to noise ratio too much.

There is also a practical reason for this choice as discussed later.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Machine Timing Synchronisation

Machine synchronisation and timestamp related functionality is handled within this module.

All LHC injectors are cycling machines. The selection of the cycle and time in the cycle to scan is a must for a meaningful size measurement and therefore the actuation has to synchronise with machine events to respect within one or two milliseconds, depending on the accelerator, the wire beam crossing time. When the user requests a certain cycle time an interrupt is programmed taking into account the measured time for the wire to fly to the centre of the vacuum chamber, according to a calibration function assigning a position in the chamber to the angle of the rotating fork. The timestamp generated by the interrupt allows posterior data correlation across modules.

Operational Bunch Profile Measurement

The transverse size measurement starts from a user request holding a few already mentioned parameters such as cycle, cycle-time, applicable harmonic number in some cases, and others such as speed and acquisition window. This module takes care of the distribution of commands and settings to the other modules for scan preparation. Once the interrupt is programmed, the unit simply awaits the notification from the other modules that the different requested data are available. When the trigger fires, the derived events will start the corresponding sequence of tasks in each of the acquiring modules. Each sequence ends in the notification of the success if the requested data is ready or, error, in case a problem was encountered, so that the error description can be retrieved. When all needed data is rendered, the correlation of the position and amplitude of the integral of each of the acquired turns takes place together with the size computation by fitting the constructed bunch profile to a gaussian.

ARCHITECTURE

The main purpose of the BWS system is to provide bunch profile and size measurements in accelerator operations. Whereas from an operational perspective the complexity of the system is hidden, the specialized usage and tuning requires access to the large amount of data and status produced while scanning by the different components of the system. Comprehensive diagnostic and testing tools need interfacing to maintain and monitor the system.

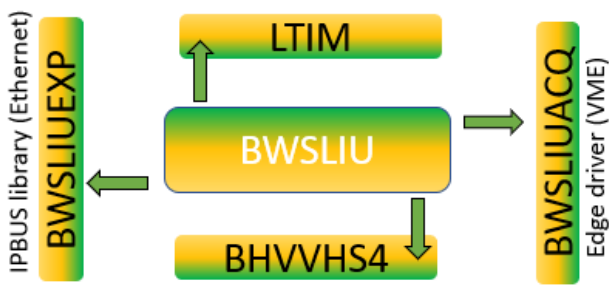


Figure 3: BWS software system event-driven architecture. Central unit FESA class with four satellite FESA classes, each representing a functional module.

The high degree of independence of the functional modules emerging from the underlying hardware design and the distinct nature of the data produced, allows the implementation of software modularisation where one coordinating unit, the operational bunch profile measurement module, orchestrates via few operation settings, the other modular components dealing with highly specific tasks. This organisation (Fig. 3) fits an event-driven architecture, using FESA classes as modules, with one central BWSLIU class having the capability of orchestrating the operations and the other satellite classes BWSLIUEXP, BWSACQ, LTIM and BHVVHS4 performing their concrete task and exposing the associated data. Such modularisation takes into account prior existence of two FESA classes, LTIM and BHVVHS4 that could be incorporated without changes only by splitting the bunch acquisition module into two, one dealing with the actual acquisition, the other handling the fine-tuning of the working point.

While FESA is very flexible and the described modular structure could be designed inside the framework, an approach to push the decoupling of the modules by using a class per functional unit adds several advantages:

- Delocalisation of modules not using VME bus, for relaxing the load of the CPU both in memory and processing.
- Interface distribution, the server is not unique anymore with higher capability of absorbing requests from the varied application set, thus, sharing the load of data publication. In this way, each set of functional users also have their encapsulated view of the system.
- Implementation in smaller units which are easier to maintain, modify and use.
- Module additions or substitutions without interference on the other units. This type of modularisation allows easy prototyping, testing, and introduction of simulation blocks when a part of the hardware is not available.
- Facilitates the test procedure. During the prototyping and testing phase multiple procedures were put in place to validate and diagnose the system during hardware commissioning and individual system tests. These can be launched without all the system being in place.

CONCLUSION

The implementation of FESA classes as building blocks in a functionally modular structure, in this case using an event driven architecture, grants high flexibility to the system in terms of prototyping and testing. It also simplifies coding structures that facilitate maintenance, encapsulates data views with extensive diagnostic possibilities and focuses interfaces adapted to the type of user exploiting them.

Following the restart of the LHC injectors the system has been deployed successfully in each of the accelerators. The commissioning was certainly facilitated by this modularisation due to the formerly described assets.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV037

REFERENCES

[1] J. Emery *et al.*, "Commissioning of the LHC Injectors BWS Upgrade", in *Proc. 10th Int. Beam Instrumentation Conf. (IBIC'21)*, Korea, Sep. 2021, to be published

[2] M. Koujili *et al.*, "Fast and high accuracy wire-scanner", in *Proc. 9th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC'09)*, Basel, Switzerland, May 2009, pp. 188–190.

[3] M. Koujili, Y. Ait-Amirat, B. Dehning, A. Djerdir, J. Emery, J. H. Alvarez, "Design of an actuator for the fast and high accuracy wire scanner", in *2011 IEEE International Electric Machines Drives Conference (IEMDC)*, May 2011, pp. 1450–1455. doi: 10.1109/IEMDC.2011.5994822

[4] R. Veness *et al.*, "Experience from the Construction of a New Fast Wire Scanner Prototype for the CERN- SPS and its Optimisation for Installation in the CERN-PS Booster", in *Proc. 4th International Beam Instrumentation Conference (IBIC2015)*, Melbourne, Australia, September 2015, 2016, TUPB061, pp. 479-482. doi:10.18429/JACoW-IBIC2015-TUPB061 https://cds.cern.ch/record/2263484

[5] J. Emery *et al*., "Design and validation methodology of the control system for a particle beam size measurement instrument at the CERN laboratory," in *2017 American Control Conference (ACC)*, May 2017, pp. 4221–4228. doi: 10.23919/ACC.2017.7963604

[6] M. Arruat, J-J. Gras, A. Guerrero, S. Jackson, M. Ludwig, J-L. Nougaret, "CERN front-end software architecture for accelerator controls", in *Proc. of International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'03),* Gyeongju, Korea, 13-17 October 2003, pp. 342-344.

[7] M. Arruat *et al*., "Front-End Software Architecture", in *Proc. ICALEPCS07*, Knoxville, Tennessee, USA, Oct 2007, WOPA04, pp. 310-312.

[8] J. Emery, P. Andersson, F. Roncarolo, and Y. Thoma, "A low fluctuation control strategy for PMSM direct drive system targeting Particle Beam Instrumentation Application" in *Proceedings, 3rd IEEE Conference on Control Technology and Applications (CCTA2019),* Hong Kong, China, August 19-21, 2019. doi: 10.1109/CCTA.2019.8920688

[9] J. L. Sirvent, J. Emery, and J. M. A. Poveda, "Design of an optical fibre based angular position sensor for wire scanners complying with ultra-high vacuum, high temperature and radiation conditions of the CERN's accelerators", 2012, https://cds.cern.ch/record/1491608

[10] C. Ghabrous Larrea *et al*, "IPbus: a flexible Ethernet-based control system for xTCA hardware", *JINST* ***10*** *(2015) no.02, C02019*. doi: 10.1088/1748-0221/10/02/C02019

[11] J. Sirvent, "Beam secondary shower acquisition design for the cern high accuracy wire scanner", Ph.D. dissertation, Barcelona University, Dec. 2018

[12] M. Gonzalez-Berges, J.O. Robinson, M. Saccani, V. Schramm, M.A. Stachon, "Test-bench Design for New Beam Instrumentation Electronics at CERN", in *Proc. ICALEPCS2019*, New York, NY, USA, Oct 2019, pp. 323-327. doi:10.18429/JACoW-ICALEPCS2019-MOPHA049

[13] S. Bart Pedersen, S. Jackson, "Graphical User Interface programming challenges moving beyond Java Swing and JavaFX", in *Proc. ICALEPCS2019,* New York, NY, USA, Oct 2019, pp. 637-640. doi:10.18429/JACoW-ICALEPCS2019-MOPHA173

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV039

A RELIABLE MONITORING AND CONTROL SYSTEM FOR VACUUM SURFACE TREATMENTS

J. Tagg, E. Bez, M. Himmerlich, A. K. Reascos Portilla, CERN, Geneva, Switzerland

Abstract

Secondary electron yield (SEY) of beam-screens in the LHC puts limits on the performance of the accelerator. To ramp up the luminosity for the HiLumi LHC project, the vacuum surface coatings team are coming up with ways to treat the surfaces to control the electron cloud and bring the SEY down to acceptable levels. These treatments can take days to weeks and need to work reliably to be sure the surfaces are not damaged. An embedded control and monitoring system based on a CompactRIO is being developed to run these processes in a reliable way [1].

This paper describes the techniques used to create a LabVIEW-based real-time embedded system that is reliable as well as easy to read and modify. We will show how simpler approaches can in some situations yield better solutions.

PROJECT AND BACKGROUND

The objective of the LESS (Laser Engineered Surface Structures) project is the commissioning of an in-situ laser surface treatment conceived to mitigate electron clouds in the Large Hadron Collider (LHC) at CERN. Secondary electrons are multiplied when they interact with the vacuum chamber walls of the accelerator and consequently form electron clouds that can negatively affect its performance.

The secondary electron emission of a surface can be reduced by surface roughening. In this project, pulsed laser processing is applied to generate micro and nanostructures on the inner vacuum chamber surface that surrounds the proton beam. In this way, secondary electrons are captured by the surface geometry. The resulting structures and the performance of the surface strongly depend on the processing parameters, such as the laser power, the scanning speed, and the line distance, as well as on the scanning pattern [2].

The final treatment must be applied in-situ in the already existing accelerator and the system must be capable of treating tens of meters of vacuum pipe autonomously. The dedicated setup to perform this is composed of a picosecond pulsed laser source and a Beam Delivery System (BDS) that shapes and couples the laser beam into an optical fiber, which guides the laser light through an inchworm robot where the beam is decoupled through a rotating nozzle (see figure 1). The translational movements of the robot are driven by a pneumatic clamping system.

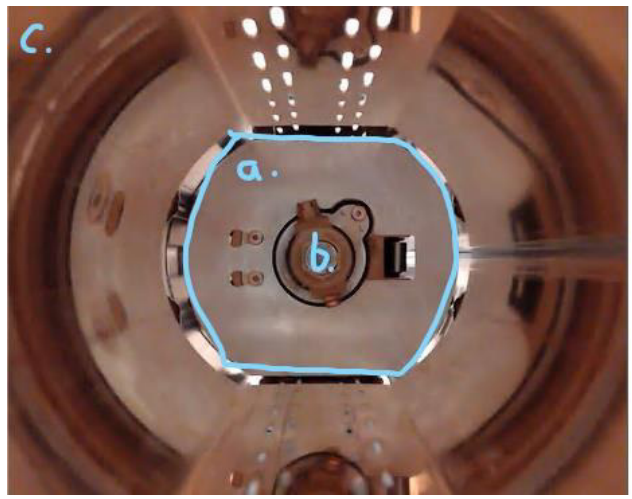


Figure 1: Longitudinal view of the inchworm inside a beam screen. The laser nozzle in the center points upwards. a. inchworm, b. nozzle, c. beam screen.

This setup requires a control system that communicates with each component and allows flexible parameter changes. The system must be reliable enough to run for many days unattended. For example, in a spiral treatment format (described later) we would need to treat 16m of beam screen while advancing by 50µm approximately every 5s. This would take up to 3 weeks. Similar times are expected for other sequences.

The system must also manage concurrent communication with all the components which make up the system and ensure that any issue is either resolved, or the system is safely stopped so the treatment can continue once the issue is resolved.

Movement of the inchworm makes up the bulk of the expected issues because of its mechanical nature. The system must be able to manage and identify movement problems and fix them where possible without affecting the overall process.

HARDWARE

The system consists of multiple hardware components connected to an NI CompactRIO (cRIO) real-time embedded system for control and monitoring. A cRIO was chosen because of successful implementations of cRIO-based control systems for other projects and because it provides a relatively straightforward programming model through LabVIEW.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV039

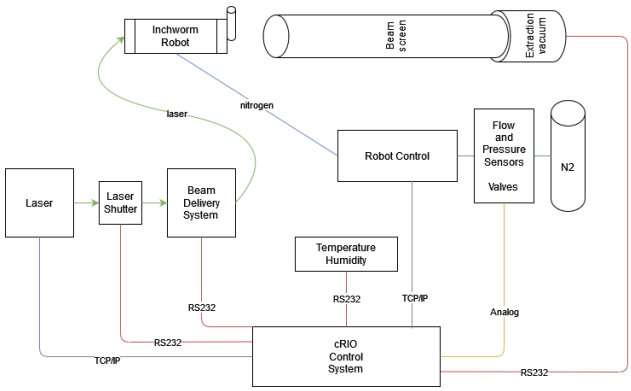


Figure 2: Main hardware topology.

Chief among the devices to control are the laser system and the mechanical robot that moves in the beam screen to distribute the laser to the surface. All hardware is connected to the cRIO through either TCP/IP connections, RS-232 or analog signals (see table 1 and figure 2).

Table 1: Connected Devices

Device	Connection
Robot	TCP/IP
Laser	TCP/IP (Telnet)
Laser shutter	RS-232
Laser BDS	RS-232
Extraction vacuum	RS-232
Temperature / humidity	RS-232
Flow and Pressure	Analog NI 9205
End switches	Analog NI 9205
Nitrogen supply valves	Relay NI 9481

The inchworm robot's movement scheme deserves some further description because it is at the core of how movements are defined in the software.

The robot is designed to move somewhat like a caterpillar and is composed of a fixed frame which contains a mobile internal sled. Both the frame and the sled can be clamped by pushing against the beam screen, which fixes its position. By activating the right clamps and moving the sled in the right direction, we can create a sequence of moves which advances the whole robot longitudinally along the beam screen.

The laser and its distribution, beyond an initial configuration of its settings, mostly turns into an on/off system. Sequences will use this facility to activate and deactivate the laser so that the treatment happens at the right times.

A nitrogen supply is used to pump nitrogen into the beam screen at the treatment point. Previous research has shown nitrogen to be an effective atmosphere for treatment of the surface [3]. Because the nitrogen tanks will run out before the full treatment is done, there are in fact 2 tanks. When the pressure from one tank falls below a defined threshold, the system automatically switches to the other tank and emails the operators so that the inactive tank can be switched out.

ARCHITECTURE

Control System Requirements

Control systems typically acquire many data points from various connected sensors and devices. Software control loops then use the acquired data to make decisions, which will affect the software itself as well as control actuators.

We decided early on that an architecture in which data would be at the center was the way to go. This leads to data being shared mostly globally in the application, and while shared data is often regarded as a danger, the small scope of the software implementation led us to accept this as a useful concept so that software procedures could more easily integrate a cross-section of application and hardware functionality.

Application Architecture

The go-to template for most LabVIEW applications over the past decade is the queued message handler (QMH). While QMHs promote modularity, and, when done properly, encourage code reuse, they can also make some applications needlessly complicated. Debugging and reading such code on embedded systems suffers from the many layers between an action from a GUI and the code that ends up running as a result. This is especially the case when frameworks force or encourage the use of re-entrant VIs, which cannot be debugged using LabVIEW's traditional debug tools, these tools being one of the main benefits of using LabVIEW. Because of this limitation and the small size of the application, it was decided to restrict the use of frameworks and to focus on a more direct and 'simple' approach to programming, in which events and their reactions are closer together.

Table 2 summarizes some of the most widely used LabVIEW frameworks, evaluating them in terms of readability, debuggability, prototyping ease and whether they can be easily instantiated multiple times.

Table 2: Framework Comparison

	Readability	Debuggability	Prototyping	Multiple instances
CVT [4]	yes	yes	yes	No
DCAF [5]	With experience	Not directly	Takes planning	Yes
QMH [6]	Can be	With experience	Nothing built-in	Yes
DQMH [7]	Lots of boilerplate code	Good testing tools	Scripting tools for quick creation of functions	Cloneable modules share some resources
Actor Framework [8]	With experience	Difficult in LabVIEW real-time	Slow to deploy	Yes

The application architecture includes a set of a few main processes (external communication, event handling, writing to file) and a series of monitoring loops which read from all the devices connected to the system. The monitoring loops mostly only read data into the system and make it available to the rest of the processes.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV039

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

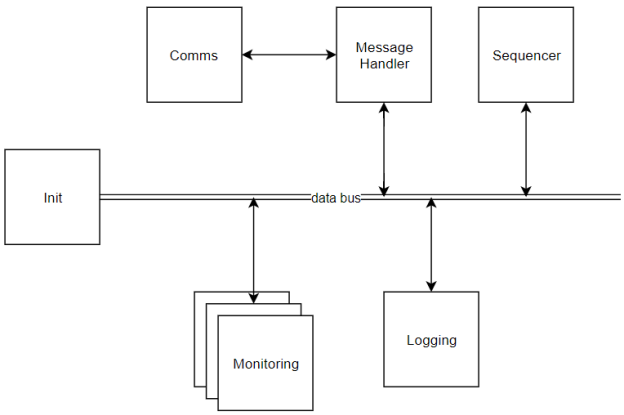


Figure 3: Software architecture.

There are few processes in the application and their roles are described here (see figure 3):

Message handler: Handles all incoming messages from the GUI. Any time the operator performs an action on the GUI, it gets sent to this process and for the most part is handled directly.

Logging: As its name implies, logs run-time data to file every few seconds so longer processes can be followed-up on in case of issues.

Monitoring: Many monitoring loops are implemented, one per device connected to the system. These processes gather all the needed data points from all devices and stores them in the central data storage for use by other processes.

Sequencer: The sequencer processes only run on-demand, and only one at a time. They simulate a series of requests and commands from the GUI so that the system can perform treatments over several days or weeks. Sequences are described in more detail further on in this paper.

IMPLEMENTATION

Because we are not using an established framework, it is important that the required attributes are built in from the beginning. The main components and attributes will be covered in this section.

Security and Safety

Like all control systems, we need to make sure that the system only does what it is supposed to. Any behavior that is outside of that established explicitly should be caught and the system should be put into a safe state. Indeed, if any behavior has not been explicitly planned for, we assume it is wrong and stop operation.

Because the movement of the robot is purely horizontal, we do not need any special consideration when stopping movement as it will simply stay in its position if we cut power to the drives.

The laser, being the active agent of this system needs to be considered more carefully because it will be damaging if it does not switch off when needed. There are multiple ways to prevent laser damage in this system. The laser can be powered off by a software command or its integrated shutter can be closed. Since a communication issue with the laser would prevent either of these safety mechanisms to be used, there is also an external laser shutter which can be independently controlled.

The system implements a safe state, which is a series of commands that is run when something unexpected happens. The main job of this routine is to stop all movement and to close all laser shutters. It puts the system into a safe mode from which it will only recover when the operator decides it is safe to do so.

Error Handling

Error handling is especially important in long-running embedded systems since operators are not monitoring the system 24/7. All errors need to be caught and, unless a recovery procedure is known, they must immediately put the system in a safe state.

Since there are many external hardware elements to be monitored during the whole treatment procedure, we make sure that communication to each device stays open and available at all times. Any communication issue immediately puts the system in its safe state.

Data Transfer and Communication

Because we wanted to keep a more direct path between events and their reactions, all processes in the application can use the central data store. This enables all processes to act in whichever way they deem necessary but introduces a higher risk of running into race conditions.

To avoid race conditions, we have a good definition of which process writes to which data point. Obviously, for data coming from external devices, only the respective monitoring loop writes that data.

All data in the application can be separated into 3 categories depending on how that data should be handled when restarting the application.

Configuration data is read-only data that comes from a configuration file on disk. This represents fixed configurations that do not change and which the application assumes will never change. We find such information as the radius of a beam-screen or the hardware address of an external device to control.

Settings represent data that the user can modify, and which can be saved to file. This allows the operator to define the details of the treatment and how the attached devices should run. These values will be remembered at subsequent launches of the application.

Run-time data is the collection of all other data the application keeps track of while it is running. Most of the data is contained here. The publishing process saves the relevant data points to a file during operation.

All 3 of these data sets are passed around the application to all processes and constitute what the application calls the data environment.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV039

Environment

The environment contains all the data needed for the various processes of the application to run properly. It is passed around as a Data Value Reference (DVR) which allows concurrent access throughout the application. One of the reasons for choosing to store this data like this rather than using existing solutions like the CVT (Current Value Table) is that this solution allows us to potentially run multiple instances of the application on the same hardware. We currently only run a single instance, but there were discussions at the beginning to run multiple systems from the same cRIO.

Giving each process access to the full dataset of the application means that each process can be more intelligent in making decisions because it has more context.

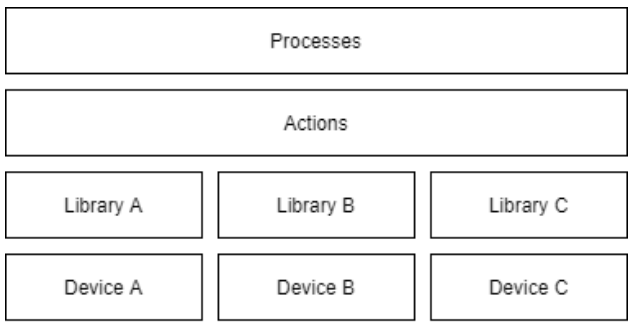


Figure 4: Application layers.

Actions

Each external hardware component is represented in the software by a library which encapsulates the communication with the device. These functions have no knowledge of the application they run in and therefore cannot be relied on to coordinate anything but the communication with that device.

To integrate these devices into a complete system, an extra layer is used above these libraries to coordinate them with each other. The first such layer is called the action layer. Each action takes in the environment data as an input and does something that the system needs to do (Figure 4). This can range from a low-level encapsulation of a single command to a single device, to a more complex one which reads data from multiple devices and coordinates their actions. The processes box represents the processes described at the end of the architecture description (monitoring, logging, communicating processes…). The processes mainly use the aforementioned actions to perform their tasks.

Sequences

To run the system autonomously, it is necessary to have some controlling process that sequences the steps necessary for a treatment run. Such processes are called sequences and they are only run on-demand when the operator sends the command. In figure 4 they run at the top-level processes category and have full access to the environment.

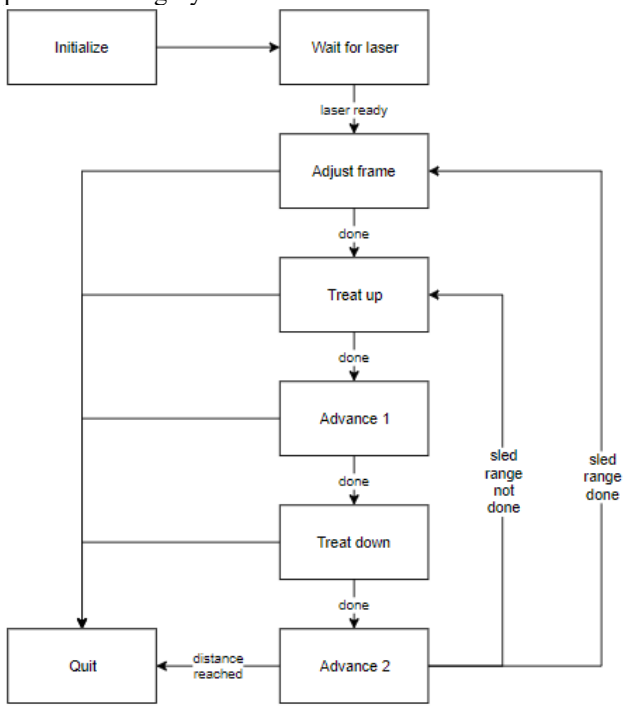


Figure 5: Line sequence logic.

More than one sequence is designed, but only one can ever run at any given time. The sequence takes control of the system and uses the data and actions to run it through the steps as the sequence defines them. When a sequence is started, the communication interface ignores most commands from the GUI because they would otherwise interfere with the sequence. In fact, sequences can be thought of as equivalent to the GUI + operator. They use the same actions as the GUI commands and simply wrap them up into a process which waits for events and reacts to them.

Sequences must also be implemented such that issues with the movement of the inchworm robot are properly handled. At each movement stage of the robot, care is taken to test for various outcomes, like getting stuck, so that it does not continue treatment.

Multiple sequences have been implemented to test their relative treatment effectiveness and are summarized here:

Spiral, as its name implies, moves the laser nozzle in a spiral fashion, in which it slowly advances while rotating the nozzle. Settings such as speed or spiral pitch allow the operator to make it more or less loose.

Line creates a movement similar to the spiral sequence, but where the forward motion is decoupled from the rotation. In this configuration it rotates with treatment, then advances without treatment, then rotates again to continue the treatment (Figure 5).

Longitudinal motion is a variation of the line sequence in which the treatment is done while moving linearly. Once it reaches the length to treat, it rotates slightly and starts treatment again.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV039

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Zigzag is a spiral sequence variation in which it does not spiral out in a single rotational direction. The rotation goes back and forth while advancing the linear stage.

Move does a simple linear move of the robot to a new position and does not involve any treatment. It is useful for movements longer than the range of the robot sled because it sequences the necessary clamping and move details.

User Interface

The user interface runs on a separate PC connected to the same network as the other devices of the system. It connects to the 'event handler' process on the cRIO to send commands to the system. The event handler simply executes the commands as they are received if no sequence is running.

Communication between the PC and the cRIO is made with the AMC (Asynchronous Messaging Communication) library [9], which allows multiple systems to send messages to each other using a UDP connection.

Commands are sent from the PC to the cRIO to execute the above-mentioned actions and sequences. All data from the environment is constantly sent from the cRIO to the PC once it is connected so that it can display the current state of the system and help the operator understand what is currently happening.

Reliability

Because the system needs to run for days at a time with minimal intervention, it is extremely important that it performs reliably. Several features of the software implementation facilitate this and were mentioned throughout the paper. The main points are summarized here for convenience.

The GUI can be connected and disconnected from the device without affecting it, thus keeping it out of the loop for the core functionality.

A safe state has been implemented which sets all hardware to a desired state. Any unplanned error puts the system in this state so that an operator or expert can check the system before running again.

Because there are so few layers of software, actions are closely linked to triggering events, leading to fewer chances for errors and mistakes to appear.

Preliminary reliability tests in the absence of full hardware integration have shown that it is able to run a sequence reliably for 2 days, after which it was stopped because it performed dozens of cycles of the sequence without issue, as indicated by the event log file. As long as the laser isn't present, a longer test won't yield more useful results at this time. Further tests are planned when the full system will be assembled.

CONCLUSION

We described the architecture and implementation choices used for the project. A custom-made data and communication structure helped simplify the overall complexity of the software so that we could focus more on direct readability and benefit from LabVIEW's built-in debugging tools.

While the system is not yet operational (not all components are ready for integration), we have benefitted from the structure of the code, allowing for quick debugging while testing integration and sequences. Because of the smaller code footprint, it is also much faster to deploy and test variations of the software while fine-tuning behavior.

It is also important to note that while this structure is used successfully in this project, we must stress that it is very likely to show its limits if functionality were to get bigger and more complex. Its application in this project was made possible by the fact that no added complexity is foreseen. Another reason why we could safely rely on a simpler architecture is that if the needs do arise, we can quite easily replace this simple structure with something more complex.

FUTURE IMPROVEMENTS

As it stands, the project is ready for integration of final components, and it is not expected that anything big will need to change. Future improvements to the system will likely be in the details and timing of the sequences, which we've modularized to a simple state machine architecture.

REFERENCES

[1] M. Sitko *et al.*, "Towards the implementation of laser engineered surface structures for electron cloud mitigation", in *Proc. IPAC 18,* Vancouver (BC), Canada, Apr.-May 2018, pp.1220-1223. doi:10.18429/JACoW-IPAC2018-TUZGBE3

[2] R. Valizadeh *et al.,* "Reduction of secondary electron yield for E-cloud mitigation by laser ablation surface engineering", Appl. Phys. Lett. 105, 231605 (2014).

[3] S. Calatroni *et al.,* " Optimization of the secondary electron yield of laser-structured copper surfaces at room and cryogenic temperature", Phys. Rev. Accel. Beams 23, 033101 (2020).

[4] https://forums.ni.com/t5/Reference-Design-Content/LabVIEW-Current-Value-Table-CVT-Library/tap/3514251

[5] https://www.ni.com/en-us/innovations/white-papers/18/introduction-to-the-distributed-control-and-automation-framework.html

[6] https://www.ni.com/en-us/support/documentation/supplemental/21/using-a-queued-message-handler-in-labview.html

[7] https://delacor.com/products/dqmh/

[8] ni.com/actorframework

[9] https://forums.ni.com/t5/Reference-Design-Content/Asynchronous-Message-Communication-AMC-Library/ta-p/3494283

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV040

A PYTHON PACKAGE FOR GENERATING MOTOR HOMING ROUTINES

A. S. Palaha, T. Cobb, G. Knap, Diamond Light Source, Didcot, UK

Abstract

Diamond Light Source uses hundreds of Delta Tau Turbo PMAC2 based motion controllers that control motors with precision and repeatability. Homing is critical to these requirements; it safely moves axes to a well-known position using a high-precision device for detection, leaving the overall system in a well-known state and ready for use. A python package called "pmac_motorhome" has been developed to generate homing routines for multiple motors across multiple motion controllers, allowing the user to write a script that is terse for standard/typical routines but allows for customisation and flexibility where required. The project uses jinja templates as `snippets' to generate the homing routine code written in Delta Tau PLC notation (PLC is the name for logic programs in Delta Tau motion controllers). The snippets can be re-ordered and grouped together, supporting the design of homing routines for multi-axis systems with mechanical limitations that require an orchestrated approach to safely home the axes. The python script using the package is kept terse using a context manager and can group axes together to the same homing group easily.

WHAT IS "HOMING" A MOTOR/AXIS?

Motors, sometimes referred to as "axes", typically turn rotational movement into moving some load. In the operation of an x-ray synchrotron and its associated beamline laboratories, there are many motors used on scales that require precise and repeatable movements.

The position of axes is typically tracked with an encoder that produces signals as the axis turns; these signals are monitored by the motion controller and converted into positions. Assuming no power loss or miscounting of encoder signals (e.g. due to speed or slippage) then the position will remain accurate. However, the counting must start from a known position, and this must be recoverable in the event of power loss or miscounting. This is where homing procedures are critical.

The known position can be provided either by a dedicated home switch, activated when the axis reaches a certain position, or one of the end-of-travel limit switches that are usually present. The manner of activation is also important; sometimes requiring moving to the switch and then in a particular direction to release the switch, or to approach the switch from a particular direction.

Another method of creating a homing signal can be to drive against a hard stop which may generate a following error (the difference between the demanded move and measured move). This is generally used for small axes that can tolerate driving into a hard stop.

Consideration must be made for axes that are coupled; that is when movement in one axis will affect another such as when they are attached to the same load. Such scenarios require homing routines that can act on multiple axes. With the possibility of many different combinations of axis types and multi-axis systems, the ability to create tailored homing routines for each axis or group of axes is a necessity.

WHY IS PMAC_MOTORHOME NECESSARY?

The first homing routine generator written in python for Diamond Light Source (DLS) was called motorhome.py. It generated homing routines as PLC code for Delta Tau motion controllers and started as a single script. As different homing scenarios became necessary, this script grew into a large monolith that was difficult to maintain and became inflexible in some scenarios. The python interface to motorhome.py could not be used to add a custom piece of PLC code that might be used for unique or rare homing scenarios.

PROJECT REQUIREMENTS

The new homing routine generator had to satisfy the following requirements to be a viable replacement for motorhome.py, and to ensure minimal disruption/effort in converting the original generator scripts into the new style:

Maintain the EPICS interface

The EPICS Input Output Controller (IOC) used to control and monitor the Delta Tau motion controller device also monitors the status of the homing routines by polling the program variables, or "p-variables". For instance, the State, Status and Group number of homing routines are accessed through $(PLC)00, $(PLC)01 and $(PLC)02 respectively; where $(PLC) is the number of the PLC program (there can be up to 32 programs stored on the motion controller). So, for the homing routine stored in PLC9, the Status of the routine would be held in p-variable 0901.

The State and Status are enumerations that indicate if a routine is operating, has completed, failed, or been aborted. The Group number indicates whether a particular axis in a group is being homed, or all axes in the group are being homed. The group is configured in the python script that imports the generator module.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Have a Python File/Script per Motion Controller or Group of Motion Controllers

The PLC code generated and stored for motion controllers at DLS is organised firstly by domain (either beamline laboratories or storage ring areas), and then by motion controller device. For example, there is a directory for a particular beamline laboratory (BLxxI) inside which there is a directory for each motion controller. The domain level directory typically contains the python script that imports the generator python module. This script configures which axes will be grouped together and how they will be homed.

Some domains have a python script per motion controller, so this must also be achieved by pmac_motorhome.

Home up to 16 Axes at once

The original homing routine generator allowed the grouping and homing of 16 axes together from one motion controller. Although rarely used in practice, this must be achievable with the new generator to maintain compatibility.

Allow Insertion of Custom Routines

Some motion setups require a specific set of homing instructions that are uniquely used, and therefore not worth adding to the generator module as a pre-defined homing type. Originally this was achieved by generating the closest matching homing routine and then modifying it manually. Though this worked, and was stored under version control, it was not automatically repeatable when the generator script was run.

The pmac_motorhome module should allow the insertion of text strings into the generated homing routine. As the typically used pre-defined homing routines would be a list of actions that would apply to motor axes, users of the python configuration script should be able to define custom sequences of actions to make a new homing routine, with custom code inserted where required.

User Friendly Python Interface

The structure of the python configuration script using the original generator, motorhome.py, was arguably simple but did not make an easily human-readable layout. Because the python configuration script is called once for each motion controller using a Makefile system, the configuration script is set out in blocks according to the name of the homing PLC to be generated, so there may be multiple blocks per motion controller. As a homing PLC is configured by adding motors to the PLC object when using motorhome.py, the configuration and grouping is done per added motor, so it is not clear which motor axes are grouped together without studying the assigned group numbers.

The pmac_motorhome generator will make use of context managers to structure the configuration script. This should group motor axes together and has all the groups for a particular PLC object listed under it, with the required python indentation providing a visually easy to understand structure. This also allows changes or configurations to be applied to motor axes, groups, or to the overall homing PLC explicitly.

Additionally, it should be possible to explicitly define a homing routine as a sequence of actions for a particular group or motor axis. For repeated sequences, such a list of actions can be defined as a separate function. The most commonly used homing sequences are provided in pmac_motorhome as pre-defined functions.

During most of Diamond's operation, it was the controls engineer's responsibility to create and maintain the PLC code for homing routines. Moving forward, this responsibility will be shared between the controls and motion engineers, combining experience and expertise in motor control and operation. This should also assist in the commissioning process of new motion axes. Having a more user-friendly interface will help the creation and maintenance of the homing routines between the two groups of engineers.

The new generator tool and interface should use python3, instead of python2 like the original generator. This should bring the tool into line with currently supported python versions and allows the use of some of the new features in the underlying pmac_motorhome code.

Can Reproduce Existing Homing Routines

To have as minimal disruption as possible during the replacement of the original homing routine generator with pmac_motorhome, a tool should be provided to convert an existing python script that imports the original generator into a pmac_motorhome style python script. This should automate the majority of the conversion process, to simplify the adoption of the new generator by controls engineers and provide confidence that existing and working homing routines will not be changed in the process. This should be demonstrated during the conversion process by comparing the outputs of the old and new generators when run for a particular beamline or domain of motion controllers.

Enforce DLS Homing Conventions

Conventions for homing PLC code at DLS includes only using PLC numbers above 8 for homing routines (1-7 are reserved for specific non-homing functions) and having all axes in the same group home with the same homing sequence.

PYTHON INTERFACE

The user interface of the pmac_motorhome generator is a python3 configuration script that imports the pmac_motorhome package. This python3 script will configure PLC objects that will render a homing routine PLC file.

The python interface uses context managers to specify the configuration of the different layers of the PLC file; the top-level object is the PLC, within this there are

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV040

groups, and within those are axes. Even solitary axes need to be defined in a group, as the group number is used to monitor homing progress and state.

In python, a context is created using the key word "with" followed by the instantiation of the object, and finally a colon; and then the next indented lines are now within the created context. Context managers allow the allocation and release of resources as needed. Boiler plate actions such as inserting a tidy-up snippet of PLC code or writing a PLC code to file can be implemented in the PLC object class. These boiler plate actions are necessary for every homing PLC, and would be executed implicitly without including it in the user level configuration script.

A simple example of a configuration script is shown in Figure 1. The editor used in the example, and in the development of this package, is Visual Studio Code.

```
from pmac_motorhome.commands import group, motor, plc
from pmac_motorhome.sequences import home_hsw

with plc(
    plc_num=12,
    controller="GeoBrick",
    filepath="/tmp/PLC12_SLITS1_HM.pmc",
):
    with group(group_num=3):
        motor(axis=1)
        motor(axis=2)

        home_hsw()
```

Figure 1: simple configuration script for a homing PLC

The use of the python language lends itself to defining custom routines that are repeatedly called. Such routines can be called inside a group to operate on some given axes, and can make use of pre-defined functions that insert blocks of code, e.g. "drive_to_limit(direction)" or "jog_if_on_limit(direction, limits)". An example of a custom routine for a group of four axes is shown in Figure 2.

```
from pmac_motorhome.commands import PostHomeMove, group, motor, only_axes, plc
from pmac_motorhome.sequences import home_hsw
from pmac_motorhome.snippets import drive_to_limit


def custom_slits_hsw(posx, negx, posy, negy):
    drive_to_limit(homing_direction=False)  # drive all slits to limit away from home

    with only_axes(posx, posy):  # home and return to limit only positive slits
        home_hsw()
        drive_to_limit(homing_direction=False)

    with only_axes(negx, negy):  # home and return to limit only negative slits
        home_hsw()
        drive_to_limit(homing_direction=False)


with plc(
    plc_num=12,
    controller="GeoBrick",
    filepath="/tmp/PLC12_CUSTOM_SLITS_HM.pmc",
):
    initial = PostHomeMove.initial_position
    with group(group_num=2, post_home=initial):
        motor(axis=1, jdist=-400)
        motor(axis=2, jdist=-400)
        motor(axis=3, jdist=-400)
        motor(axis=4, jdist=-400)

        custom_slits_hsw(posx=1, negx=2, posy=3, negy=4)
```

Figure 2: example of custom homing routine definition

Custom code snippets can also be introduced, and with the control of the sequence of pre-defined and custom blocks even more control of the customisability of homing sequence generation is possible.

Finally, the layout and indentation due to the use of context managers creates a more readable configuration script; where groups of axes are visually one block of code. The naming of the snippet functions and routines aims to be as descriptive as possible, while the availability of pre-defined homing sequences for the most typical and simple of homing scenarios will allow for terse and concise configuration scripts for most homing scenarios.

GENERATING PLC CODE

The python configuration script creates a PLC object that contains groups of axes, and all these objects are instances of classes that contain data members detailing the homing sequence to be performed. These PLC objects are fed to the homing sequence PLC generator.

The pmac_motorhome generator compiles the homing routine PLC code by piecing together blocks of code formed from code templates. The Jinja engine is used to interpret and fill in the templates. The templates contain special place-holders allowing python-like code to be inserted and eventually rendered into text strings. These templates are referred to as "snippets" in the pmac_motorhome package, and generally correspond to functional blocks of code, such as the "drive_to_limit(direction)" function, among others.

The interpreted code in the templates allows loops to be performed, and more snippets to be invoked and inserted in place, such as for multiple motors in the same group. It also provides conditional logic, allowing for snippets to be included if, for example, an option to perform a move after homing is supplied. When invoking a particular snippet with this option, the corresponding code would be rendered, otherwise it would be omitted. The snippet shown in Figure 3 shows the post home action snippet, using an if-statement to control whether the code template is rendered or not. If the python class member group.post contains a non-empty string, the template will render.

```
{% if group.post and group.post != "" %}
{% include "debug_pause.pmc.jinja" %}

;---- PostHomeMove State ----
if (HomingStatus = StatusHoming or HomingStatus = StatusDebugHoming)
    HomingState=StatePostHomeMove
    ; Execute the move commands
    if (HomingStatus = StatusHoming or HomingStatus = StatusDebugHoming)
        {{ group.post }}
    endif
endif

{% endif %}
```

Figure 3: snippet inserting code for post home actions

The snippets are structured in such a way as to represent simple and broadly singular functions, for the benefit of developers and maintainers of the code and to correspond to the required structure of the homing PLC code that already exists at DLS for its Delta Tau motion controllers. A particular snippet will usually perform one function, however there is a base snippet that includes all the beginning and ending boiler plate, and contains loops for all the groups and axes contained in the PLC.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The final step is writing the rendered code into a file, the path for which is specified in the configuration of the PLC object.

CONVERTING TO THE PMAC_MOTORHOME GENERATOR

A converter tool is provided in the package to convert the original python configuration scripts in the DLS motion directories to the new style of python script using the pmac_motorhome package. This is to facilitate the adoption of the new style of homing code generation with as little intervention as possible, and to check that the new generator can produce the same homing PLC code as that which already exists.

The converter tool (written in python3) operates on a domain level or motion controller level directory and creates two copies of these directory structures in a temporary location: old and new. The old copy has the original generator script called through a subprocess call in a python2 shell (as the original generator is a python2 tool).

The new copy is also run through a python2 shell with the original generator script, but the generator code is "shimmed" (or substituted) with code that records the data structures produced without creating a file on disk. These data structures are the PLC objects that contain group and axis objects and are required in order to create a pmac_motorhome style configuration script. As the converter tool is run in python3 and the shimmed motorhome.py generator is run in a separate python2 shell, the PLC data object must be loaded in some way into the python3 instance of the converter tool. This is achieved by creating a FIFO (First-In-First-Out) pipe file from inside the python3 converter tool. Then the shimmed motorhome.py generator in python2 will write to that FIFO file a copy of the PLC data objects it creates.

To ensure the data is compatible between python2 and python3, the PLC data object is first serialized into a byte stream using the pickle package before being encoded using the struct package into a packed form. This encoded byte stream is written to the FIFO file, where it is read out by the converter tool in python3 that is monitoring it. The byte stream is unpacked and then un-pickled to extract the native python data structure of the PLC object. Using this PLC object, the configuration script using pmac_motorhome can be generated in python3 and run in its own python3 shell to create the homing sequence PLCs in the new copy of the motion directories.

The final step of the conversion process is to compare the homing PLCs created using motorhome.py with those created using pmac_motorhome; this is done with a call to the "diff" command with options to ignore blank lines and space changes. The result is then reported to the terminal indicating whether any comparisons or conversions failed, and provides a short, generated script to copy the new python3 configuration script to the original motion directory for the beamline or motion controller, if the user chooses to do so.

It is expected that not all conversions will be successful, however enough development has been done to ensure that about 85% of existing DLS motion homing directories will convert successfully, and of the remaining directories the majority of individual homing PLC files will be successfully converted. The remaining homing PLCs that were not converted can then be investigated manually, the expectation being that most will be custom PLCs or edge cases that will require some tweaking of the configuration script.

CONCLUSION

The pmac_motorhome package has been specified, designed and developed with the aim of improving the tool to generate critical motion homing routines at DLS, and other institutions that use Delta Tau motion controllers. The improvements include a more friendly and readable user interface, a better structured code base that is more maintainable and a testing framework with continuous integration built in. Documentation is generated in a web page format with explanations, how-to guides, tutorials and references; this will be beneficial to developers who want to add another defined homing routine or new snippets. Consideration has been given to the roll-out and adoption of the new tool, with effort put into removing any barriers to adoption and creating a conversion process that is as simple and confidence inspiring as possible for the controls engineers as the targeted users.

This tool provides the means to create customised homing sequences where required, and always be able to recreate it without manual intervention or editing. Finally, it will and allow better collaboration between motion and controls engineers through its more readable user interface and documentation.

It is hoped that pmac_motorhome may prove useful to other institutions using many Delta Tau motion controllers that require homing routines.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV042

COLLISION AVOIDANCE SYSTEMS IN SYNCHROTRON SOLEIL

C. Engblom†, S. Zhang, S. Bouvel[1], D. Corruble, G. Thibaux, S. Akinotcho, P. Monteiro, L. Munoz, B. Pilliaud[2], L. Amelineau, Synchrotron SOLEIL, St. Aubin, France
[1]also at EFOR, Paris, France
[2]also at Mediane Systems, Paris, France

Abstract

Beamlines at Synchrotron SOLEIL are finding that their experimental setups (in respect to their respective sample environments, mechanical systems, and detectors) are getting more constrained when it comes to motorized manoeuvrability - an increasing number of mechanical instruments are being actuated within the same workspace hence increasing the risk of collision. We will in this paper outline setups with two types of Collision Avoidance Systems (CAS): (1) Static-CAS applications, currently being employed at the PUMA and NANOSCOPIUM beamlines, that use physical or contactless sensors coupled with PLC- and motion control- systems; (2) Dynamic-CAS applications, that use dynamic anti-collision algorithms combining encoder feedback and 3D-models of the system environment, implemented at the ANTARES and MARS beamlines but applied using two different strategies.

INTRODUCTION

System actuation in small or limited workspaces can be a delicate matter when taking the risk of collision into consideration. Synchrotron multi-techniques experimental environments are becoming more difficult in this matter as they combine complex beam-focusing setups, sample stages, and detectors – each section often actuated in many Degrees-Of-Freedom (DOF), sometimes with overlapping workspaces, and each section often very fragile and expensive/time-consuming to repair.

The traditional (and most direct) approach to this problem is to introduce workspace limitations (with mechanical hard-stops and/or limit switches to the various actuators) to different subsections, hence assuring non-overlapping workspaces and thus eliminating the risk of collision. This method is however only limited to static (e.g. unchanging) and less constrained environments in the sense that the setup is set in a fixed configuration and no additional systems should be introduced into the workspaces, nor that the different workspaces should ever overlap.

This paper will outline four motorised Collision-Avoidance-Systems (CAS) at SOLEIL that have been adapted to *dynamic or complex workspaces*, particularly where overlapping workspaces are being used. The CAS applications are here classified as:

1. **Static-CAS**: Systems that use proximity- or touch-based sensors coupled with PLC- and motion control-systems.
2. **Dynamic-CAS**: Systems that use motion controllers with integrated dynamic anti-collision algorithms combining encoder feedback and 3D-models of the system environment to avoid collisions.

† christer.engblom@synchrotron-soleil.fr

STATIC-CAS AT THE PUMA BEAMLINE

PUMA [1] is an ancient materials analysis beamline optimised for 2D- imaging with hard X-rays in the 4-23 keV range. The beamline offers its users a range of analytical tools in the form of X-Ray fluorescence (XRF), absorption spectroscopy (XANES), and powder diffraction (XRD). A second experimental stage will be added in the future for 3D imaging with up to 60 keV X-ray beam energy.

This section will focus on the setup situated in the CX-hutch where its beam-focusing section, sample-stage, and detector-support all move in collision-range of each other.

PUMA Experimental Station Overview

The PUMA CX Environment system consists of two motorised table platforms: one holds the Kirkpatrick–Baez (KB) mirror subsystem, its sample goniometer stage, microscope and XRF detectors, while the other table supports the 2D X-ray camera detector. Figure 1 illustrates the overall setup, here each subsystem annotated with the DOF used in the CAS. In total, the complete system holds 41 motorised axes - with overlapping workspaces over 16 DOF for the mirror-chamber, sample stage, and detectors.

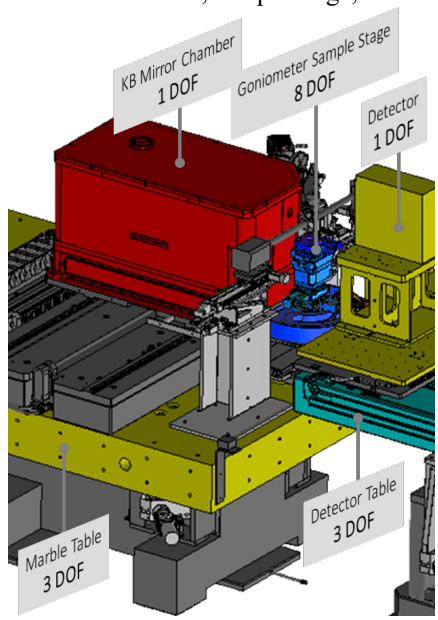


Figure 1: PUMA CX environment overview to be controlled with a CAS, here portraying the five subsystems with a total of 16 degrees of freedom (DOF) used in the CAS.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PUMA Experimental Station Control & CAS Architecture

Figure 2 shows the control architecture for the complete system implemented in the CAS: low-level control is applied via a SOLEIL-standardised 4-controller(ControlBox)-driver(DriverBox) setup [2] and a PLC system, and high-level control is done via the TANGO framework [3].

The CAS framework is introduced at the low-level, with proximity-sensors and a safety light curtain sensor strategically installed at collision risk-zones. A PLC continually monitor and filter these sensor states and blocks (via dedicated PLC-controller TTL signals, here marked 'Inhibition' in Fig. 2) specific axis movements if it deems a collision is imminent. All controllers contain application-specific microcode to take the appropriate action based on these PLC-signals. To 'unblock' motors, the user can apply an 'Acknowledge'-command to the PLC via the TANGO framework, which then transfers it to the controllers via TTL-signals and hence unblocks the motor movements in the appropriate direction. The *Inhibition* and *Acknowledge* signals have been made active so that if connectivity or power-loss issue arise, the motors are blocked by their controller.

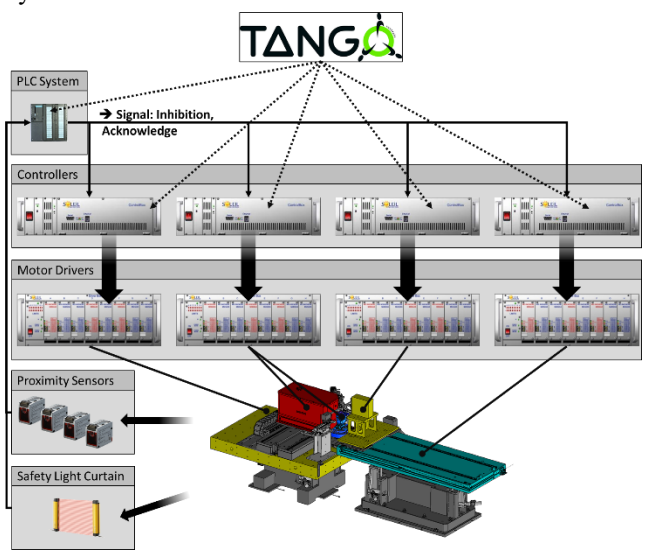


Figure 2: PUMA CX CAS control architecture, the SOLEIL standard controllers (ControlBox [1]) here complemented with PLC-treated signals from proximity- and safety light curtain sensors.

Each anti-collision sensor outputs two states (collision/no-collision) to the PLC system: the safety light curtain sensor simply does this by detecting laser-breaks in the visible path, the proximity sensors must however be preconfigured to output their collision-states if the measured distance falls within certain thresholds. The output from the proximity sensors can however give off false activations (in the form of ON/OFF flickering) and are therefore filtered in the PLC system for more robust use. The PLC then blocks/unblocks specific axis movements using the combined sensor data using pre-filled table-data for reference.

The whole CAS can be disactivated via the TANGO device server with a password protection.

STATIC-CAS AT THE NANOSCOPIUM BEAMLINE

The NANOSCOPIUM beamline [4] is dedicated to hard X-rays (5-20 keV) fast scanning multimodal and multiscale (35 nm to 1 µm) 2D- and 3D imaging with high spatial resolution (nanometric). The beamline offers cutting-edge X-ray nanoprobe imaging and tomography techniques. It provides a wide range of complementary imaging and spectroscopy modalities, to which coherent diffraction imaging is associated.

The NANOSCOPIUM CX2 and CX3 nanoprobe stations are in exploitation in sequential mode: CX2 is based on FZP and CX3 is based on KB nanofocusing optics. CX2 offers Ptychography with highest spatial resolution down to 35 nm and full field X-ray microtomography in absorption and phase contrast modes. For which many different types of motorized detectors and optical elements are installed in a very compact space, each can be inserted or extracted to build a specified station environment according to the experimental requirements. However, the risk of collision between different elements is increasing and needs to be treated for its safe operation.

NANOSCOPIUM CX2 Overview

The general setup of the CX2 environment is shown in Fig. 3, where the sample stage and most of the detector sections are installed in close proximity to each other. In addition, the Fresnel Zone Plate (FZP) and Central-Stop (CS) optical stages may also approach the same area and possibly collide with the vertical support marble during their displacement, which complicates the issue even further.

The risk of collision doesn't only present itself from the motorised elements: users should be able to manually displace certain sections (e.g.: the XRF stations along a circular rail) and possibly install/uninstall beam transfer tubes that approach the sample environment.

The first CAS control has been implemented to take care the collision risks between the optics stage, 3 detector stages, the support marble and the CX3 beam transfer tube. It will be completed by an evolution study in the future to take account collision risks with the sample stage and XRF detector stages.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV042

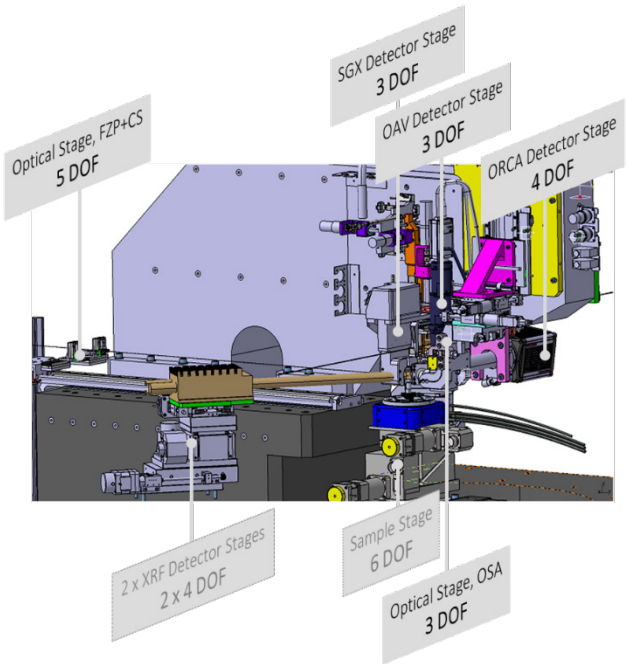


Figure 3: NANOSCOPIUM CX2 environment overview to be controlled with a CAS system, here portraying the five subsystems with a total of 18 degrees of freedom (DOF). Note that the two XRF-stations and Sample Stage are not yet included in the CAS.

NANOSCOPIUM CX2 Control & CAS Architecture

Figure 4 shows the general architecture that is currently in place: low-level control is using a high-performing SOLEIL-standardised 4–controller (Powerbrick LV [2][5] & Controller+Piezo driver) setup [2] and a PLC system. All the controllers (1 master and 3 slaves) are synchronized by a fibre-optic ring network (MACRO ring). High-level control is exerted via the TANGO framework [3].

The NANOSCOPIUM-CAS operates primarily in the PLC system and continually receives: the states of all the CAS-limit switches, each controller state (ex: running/not-running), and the general locations of the subsystems (based on encoder feedback). The PLC system also relies on user-specified (via TANGO devices) experimental configurations – where each configuration defines a set of optical stages and detectors to be used in the workspace. The PLC then authorizes/refuse stage movements depending on the combined sensor- and user-configuration- data with pre-filled reference tables.

All controllers contain specifically written microcode to transfer its running state to the PLC – only the master controller has additional CAS programs to take the appropriate action based on commands from the PLC. In addition, the master controller also holds automated scripts to safely insert/extract the subsystems into use.

Currently under investigation, the PLC-Controller-GPIO communication could be replaced by a serial communication bus (ex: EtherCAT) – which would significatively simplify the communication architecture.

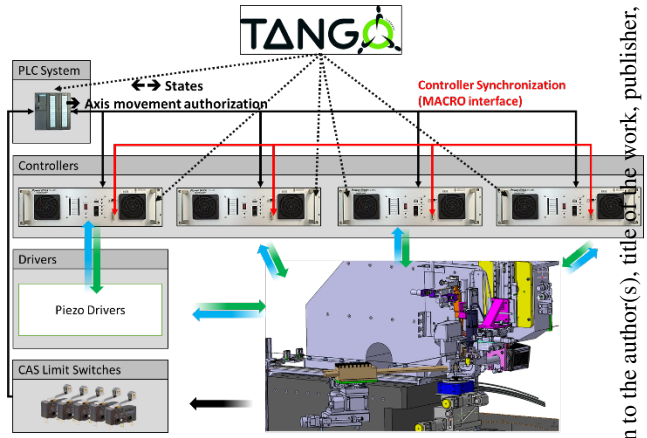


Figure 4: NANOSCOPIUM beamline CAS control architecture, the SOLEIL standard high-performance controllers (Powerbrick LV [2][5]) here complemented with a PLC-system and CAS limit switches (green arrows indicate control signals, blue arrows are encoder feedback signals, black are state/command data).

DYNAMIC-CAS AT THE MARS BEAMLINE

The MARS beamline [6] seeks to research radioactive matter in the fields of biology, chemistry, and physics. The beamline provides users amongst other things with characterizations with transmission and high-resolution X-ray powder-diffraction (XRD), Wide Angle X-ray Scattering (WAXS), and Small Angle X-ray scattering (SAXS).

The MARS Detector Support was installed in 2020 in the CX3 experimental station to actuate heavy detectors ($\lesssim$ 50kg) for SAXS experiments and would be doing so using relatively large movements in the sample stage vicinity (and would therefore induce risk of system-collisions).

MARS Detector Support: Overview, Control- and CAS Architecture

Figure 5 shows the overall system overview annotated with its DOF. The system only contains 5 motorised axes, of which the three axes (*TS*, *TX*, *RX*) are the ones running the risk of system-collision with its surrounding environment. As is seen in the same figure, the control- & CAS architecture is relatively simple and direct; a single high-performing SOLEIL controller [2][5] is used which is connected to a high-powered (> 1 kW) motor amplifier for the rotational axis. The CAS algorithm here is generated from the system 3D-models and is entirely implemented in the controller which will dynamically calculate the (*TS*, *TX*, *RX*) motor *software limits* in function of their respective encoder values to prevent their entries in collision areas. This constant re-calculation of the motor software limits allows for a more complex and detailed workspace - assuring that movements on the (*TS*, *TX*, *RX*)-axes would never cause a system-collision with a perceived virtual object in its vicinity.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

These virtual objects are the core of the MARS dynamic-CAS algorithm and it is essential that they mirror the real environment.

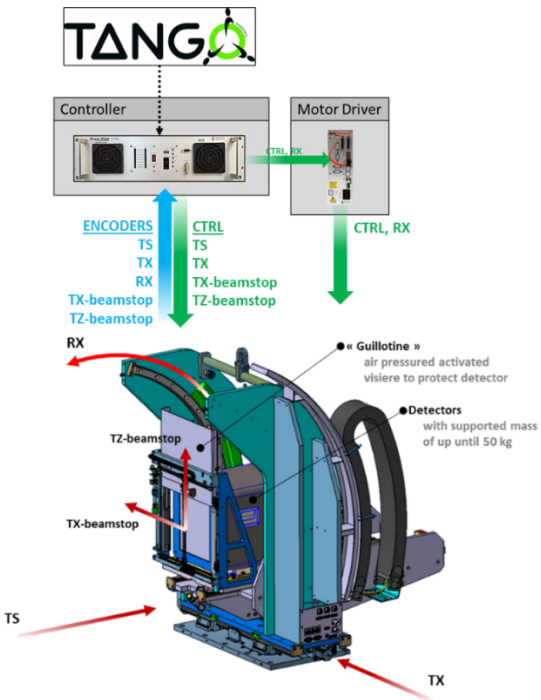


Figure 5: MARS Detector Support overview and control architecture - here using a SOLEIL standard high-performance controller (Powerbrick LV [2][5]) matched with a high-powered driver for the rotational axis.

MARS Detector Support: dynamic-CAS algorithm

Figure 6 illustrates the five-step approach in which the MARS dynamic-CAS was implemented – starting here with system 3D-models (and its environment), extracting the necessary kinematic- & spatial data, identifying collision points (using MatLab simulations), and then generating and implementing the CAS-algorithm. We will in this paper focus on the second and third point: *Simulations & Collision Detections*, and *Anticollision equation generation*.

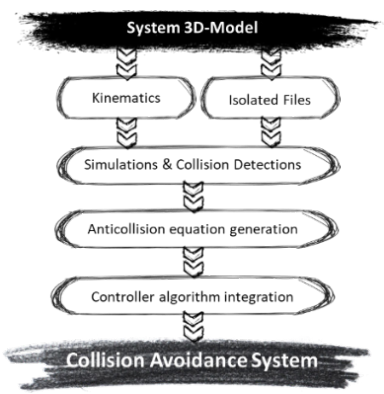


Figure 6: The five-step approach used in implementing the MARS Detector Support dynamic-CAS.

Dynamic-CAS: Finding Collision Points & Generating the Algorithm

Figure 7 shows the approach taken to generate the CAS algorithm. The 3D-models of the Detector Support system and its environment were implemented (with the system kinematics) into Matlab where a batch of simulations would move the system axes to find all collision points with its environment. These simulation-movements would virtually test all points (and map them) in the (*TS*, *TX*, *RX*)-space (with pre-defined small step increments) for collisions.

To further condense the collision point data (which is in this case 3-dimensional), and make it more suitable for controller implementations, linear collision boundaries were calculated - thus grouping together collision data-points using linear equations.

The line equations (depicting the collision boundaries) could then all be transformed into a set of geometrical closed half-space equations (see Fig. 7). This set of equations (each equation stretching over n dimensions, where n is the number of actuators used), would essentially contain as many equations as there were boundaries – making it scalable and simple enough to implement in the controller.

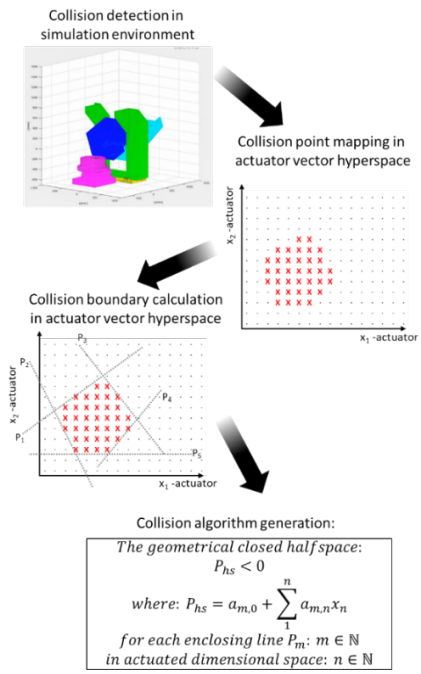


Figure 7: Finding collision point data and generating the dynamic-CAS algorithm on the MARS beamline.

DYNAMIC-CAS AT THE ANTARES BEAMLINE

The ANTARES beamline offers its users a spectroscopic non-destructive nanoprobe (k-space nanoscope) that is used to study materials using Angle Resolved PhotoEmission Spectroscopy with nanoscale lateral resolution (nanoARPES) [7].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV042

The ANTARES experimental station (see Fig. 8) is set in a vacuum chamber and contain several piezo-driven stages. Three of these stages: the Order-Sorting-Aperture (OSA), the Fresnel-Zone-Plate (FZP), and the Sample Stage have their workspaces overlap and run the risk of collision when in use.

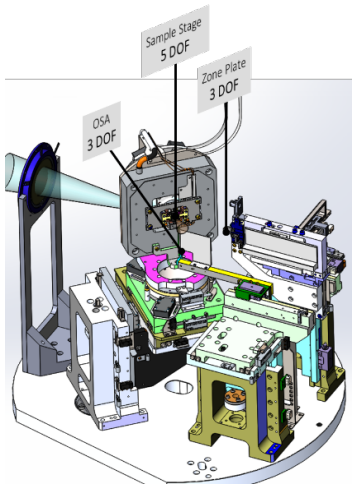


Figure 8: ANTARES Experimental Station overview, here showing the 11DOF that are used in the CAS.

ANTARES Experimental Station: Overview, Control- and CAS Architecture

The experimental station makes use of a two-controller configuration (see Fig. 9): two synchronized Delta Tau Controllers [2], both using Piezo- and stepper-drivers to interface with the motors and actuators. The control-CAS system controls 11 DOF in total.

The ANTARES beamline makes use of a dynamic-CAS – here essentially blocking\unblocking motor axes in function of the system encoders. It does this by identifying what specific region each subsection finds itself and blocks other stages from entering the same space.

CONCLUSION

The trend towards multi-techniques & multimodal beamlines makes for more densely packed experimental stations with overlapping (or more complex) workspaces – this makes Collision Avoidance Systems a more pertinent addition to control-systems. As is shown in this paper, the CAS-architecture and approach differ somewhat between applications but the common thread in CAS would indicate a closer collaboration between metrology, mechanical design, and electronics- & control- architecture.

Advanced-CAS development is a part of the roadmap for the SOLEIL electronics group to address future synchrotron beamline needs in respect to: workspace limitations, and experiment automatization (in particular integrating industrial arm robots in the mechatronic architecture).

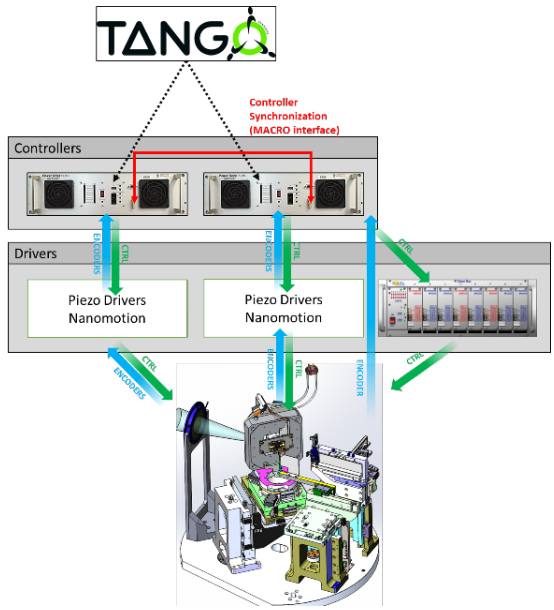


Figure 9: ANTARES Experimental Station Control- & CAS architecture, here with two synchronized SOLEIL standard high-performance controllers (Powerbrick LV [2][5]) using piezo- & stepper- drivers to control the 11 DOF. The Controllers contain the anti-collision algorithm.

ACKNOWLEDGEMENTS

The authors would like to thank: the beamline teams, mechanical engineers and metrologists who were involved in these developments, as well as all SOLEIL staff who have aided or participated in any way to the development of the CAS systems.

REFERENCES

[1] PUMA beamline, https://www.synchrotron-soleil.fr/fr/lignes-de-lumiere/puma

[2] S. Z. Zhang, Y.-M. Abiven, F. Blache, D. Corruble, C. K. Kheffafa, and S. M. Minolli, "Revolution Project: Progress in the Evolution of Soleil Motion Control Model", in Proc. ICALEPCS'15, Melbourne, Australia, Oct. 2015, pp. 201–204. doi:10.18429/JACoW-ICALEPCS2015-MOPGF04

[3] Tango Control Systems, https://www.tango-controls.org/

[4] NANOSCOPIUM beamline, https://www.synchrotron-soleil.fr/fr/lignes-de-lumiere/nanoscopium

[5] Powerbrick LV (Faraday Motion Controls), https://www.faradaymotioncontrols.co.uk/page/power-brick-lv-ims/

[6] MARS beamline, https://www.synchrotron-soleil.fr/en/beamlines/mars

[7] ANTARES beamline, https://www.synchrotron-soleil.fr/en/beamlines/antares

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV046

MODIFICATION OF DATA ACQUISITION SYSTEM IN HLI-II EXPERIMENTAL STATION*

Zhen Zhang†, Gongfa Liu
University of Science and Technology of
China National Synchrotron Radiation Labor-atory, Hefei, China

Abstract

With the proposal of the concept of super-facility in recent years, users of experimental stations only need to pay attention to data with scientific significance, and the management of massive experimental data are assisted by the super-facility technical support platform to effectively improve user efficiency [1]. Based on this theory, we modified the data acquisition system of the XMCD experimental station in HLS-II. We continue to use LabVIEW software to reduce development workload. Meanwhile, we have added the interaction program with the high-level application in the original data acquisition process under the principle of keeping the user habits of XMCD experimental station. We have modularized the XMCD experimental software and redesigned the experimental architecture into 4 modules: Swiping Card Module, Experimental Equipment Control Module, Storage System Interaction Module and Data Management System Interaction Module. In this way, we have completed the collection of rawdata and metadata, the docking of the data persistent storage system, and the docking of data centralized management.

INTRODUCTION

As a synchrotron radiation light source, Hefei Light Source (HLS-II) provides a basic research platform for multi-disciplinary research on cutting-edge topics [2]. Under the important trend of informatization construction [3-5] of large scientific equipment at home and abroad, we have upgraded the data acquisition system of the HLS-II XMCD experimental station.

ARCHITECTURE

Based on LabVIEW experimental system, the data acquisition system (DAQ) of HLS-II XMCD experimental station can accomplish the mission of scientific data acquisition. However, under the information construction trend of centralized data management, experimental data needs to be uploaded to the experimental data management system together with experimental metadata. At the same time, in order to facilitate the management of data, relevant information of the experimenters also needs to be collected. Therefore, based on the development of the existing XMCD experimental station data acquisition system, we upgraded it to meet the above requirements. The acquisition system architecture is shown in the Fig. 1 below.

The data acquisition system is divided into four modules: swiping card module, experimental equipment control module, storage system interaction module and data management system interaction module. After the user swipes the ID card, the data

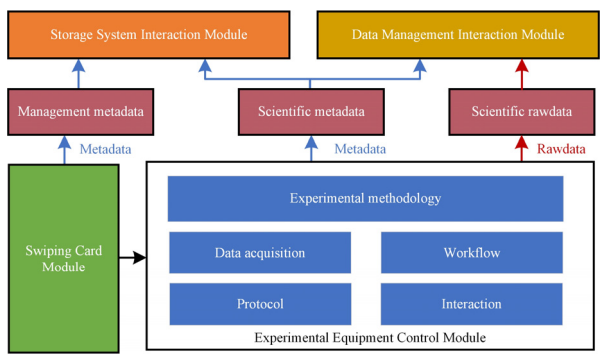


Figure 1: The architecture of DAQ system.

DATA ACQUISITION

User Management Metadata Acquisition

The User management metadata is collected from the HLS-II user platform. The process is shown in Fig. 2: (1)User needs to swipe the ID card at the card reader; (2)The user authentication module sends the collected ID number to the HLS-II user platform via RESTful API, and waits for the HLS-II user platform to return the verification result; (3)If the verification is successful, it means that

* Work supported by Key R & D Project of Hefei Science Center, Chinese Academy of Sciences (No.2019HSC-KPRD003)
†zznsrl@mail.ustc.edu.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV046

when the user has applied for the current experimental machine of the experimental station, scientific experiments can be carried out, otherwise the card needs to be swiped again; (4)After the user checks the information, the management metadata will be output.

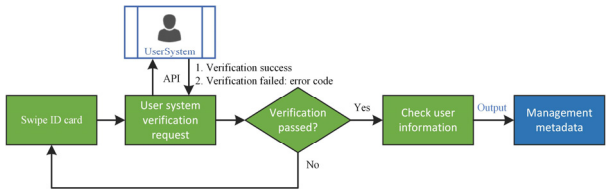


Figure 2: User authentication process.

Scientific Rawdata Acquisition

The original experimental data of the XMCD experimental station comes from the KEITHLEY 6517B electrometer. It contains two parts of data: the electrometer data at the gold mesh in front of the sample and the electrometer data at the sample. Combined with the energy point data, the three together form the the scientific rawdata. These data will be packaged into HDF5 format files together with the scientific metadata. The HDF5 file is shown in Fig. 3.

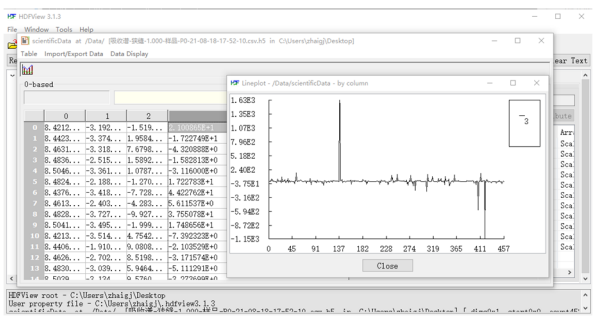


Figure 3: Scientific rawdata.

Scientific Metadata Acquisition

Combined with the management requirements of the experiment management platform and XMCD experiment station, we have compiled the relevant metadata of the XMCD experiment which is shown in Table 1.

Table 1: Scientific Metadata

metadata	sources
beamCurrent	beamline
beamPosition	beamline
gasComposition	beamline
gasPressure	beamline
waterFlow	beamline
waterPressure	beamline
waterTemperature	beamline
lowTemperature	beamline
groundResistance	beamline
vibration	beamline
magneticFieldStrength	beamline station
vacuum	beamline station

DATA MANAGEMENT

Storage System Interaction

HDF5 [6] is a common cross-platform data storage file, which can store different types of images and digital data, and can be transmitted on different types of machines. At the same time, there is also a function library for unified processing of this file format. HDF5 files generally contain two parts: groups and datasets. Groups are similar to folders and serve as the root directory of datasets. Datasets contain rawdata and metadata. Rawdata is the core data of HDF5 files, and metadata is used to store attribute values related to the rawdata. In Hefei light source, the original data of HDF5 files are mainly used for the preservation of experimental data, such as photoelectron spectroscopy.

After storage system interaction module gets the scientific rawdata and the scientific metadata transferred from the experimental equipment control module, it encapsulates them in HDF5 format. The scientific rawdata is the rawdata of HDF5 file and the scientific metadata is the metadata of HDF5file. Then the module generates the md5 checksum file of the hdf5 file. Through the API provided by the storage system, these files will be uploaded to the specified directory of the storage system

Data Management System Interaction

JSON [7] is a lightweight data exchange format. It is based on a subset of ECMAScript [8] (the js specification formulated by the European Computer Association), and uses a text format completely independent of the programming language to store and represent data. The concise and clear hierarchical structure makes JSON easy to read and write, easy to parse and generate by machine, and effectively improve network transmission efficiency.

According to the requirements of the scientific data management system, the management metadata and experimental metadata collected during the experiment need to be filtered and flattened by the data management system interaction module, and packaged into a single-layer JSON file which is shown in Fig. 4.

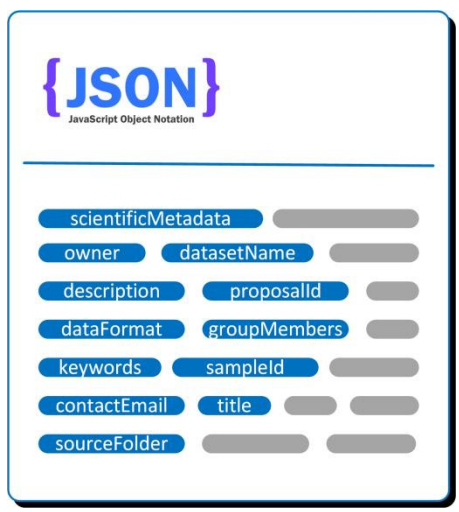


Figure 4: JSON file of metadata.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV046

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONCLUSION

In response to the need for centralized data management, we upgraded the data acquisition system of the XMCD experimental station in Hefei light source. The upgraded data acquisition system can better realize the collection of scientific rawdata, scientific metadata and management metadata. At the same time, the data acquisition uploads the collected experimental data to the storage system and the experimental data management system to provide a basis for future experimental reproduction and high-level software data processing.

ACKNOWLEDGEMENTS

Thanks to Hefei Light Source BL12B beamline station for cooperating with the upgrade of data acquisition system.

REFERENCES

[1] Wang Chunpeng, Yu Feng, Liu Yiyang, Li Xiaoyun, Chen Jige, Thiyagalingam Jeyan, Sepe Alessandro, "Deploying the Big Data Science Center at the Shanghai Synchrotron Radiation Facility: the first superfacility platform in China", Machine Learning: Science and Technology, 2021,2(3). doi:10.1088/2632-2153/abe193

[2] Lin W. *et al*., "The Upgrade Project of Hefei Light Source (HLS)[C]", in *Proc. International Conference on Intelligent Information Processing*, Oct. 2010.

[3] Blaiszik B. *et al*., "The Materials Data Facility: Data Services to Advance Materials Science Research", JOM, 2016, 68(8):2045-2052.

[4] B. V. Luvizotto, K.A., E. Stavitski, H. Bassan, "XLive: Data Acquisition and Visualization at the NSLS-II ISS Beamline", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 962-965. doi:10.18429/JACoW-ICALEPCS2017-TUPHA211

[5] Blaiszik B. *et al*., "Data automation at light sources", in Proceedings of the 13th International Conference of Synchrotron Radiation Instrumentation (SRI2018), https://doi.org/10.1063/1.5084563

[6] Dheeraj Joshi, Krima Shah, Harshan Andrews, Lakshmi Kurup, "Fast Analysis & Storage of Big Data Using HDF5".

[7] John Resig, "Pro JavaScript™ Techniques", Apress:2007-01-01.

[8] Nicholas C. Zakas, "Understanding ECMAScript 6", Network Security, 2016, 2016(12).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV047

CONTROLLING THE CERN EXPERIMENTAL AREA BEAMS

B. Rae*, V. Baggiolini, D. Banerjee, J. Bernhard, M. Brugger, N. Charitonidis, M. Gabriel, L. Gatignon, A. Gerbershagen, R. Gorbonosov, M. Hrabia, M. Peryt, G. Romagnoli, C. Roderick
CERN, 1211 Geneve 23, Switzerland

Abstract

The CERN fixed target experimental areas are composed of more than 8 km of beam lines with around 800 devices used to define and monitor the beam parameters. Each year more than 140 groups of users come to perform experiments in these areas, with a need to control and access the data from these devices. The software to allow this therefore has to be simple and robust, and be able to control and read out all types of beam devices. This contribution describes the functionality of the beam line control system, CESAR, and its evolution. This includes all the features that can be used by the beam line physicists, operators, and device experts that work in the experimental areas. It also underlines the flexibility that the software provides to the experimental users for control of their beam line, allowing them to manage this in a very easy and independent way. This contribution also covers the on-going work of providing MAD-X support to CESAR to achieve an easier way of integrating beam optics. An overview of the on-going software migration of the Experimental Areas is also given.

INTRODUCTION

The CERN experimental areas are a complex system of beam lines and beam intercepting devices that are able to provide a large variety of different particle beams to different experiments and detector assemblies. They serve both fixed target experiments and test beams [1]. The most important aspect of these unique experimental facilities is the possibility for experimental users to control and to monitor beam parameters from dedicated terminals installed in their respective control rooms. Such parameters include the access to the experimental zones, the beam intensity via collimator settings, the magnet currents, which are defining the beam trajectory and focal properties, the particle species via the use of targets, converters and absorbers, and the instrumentation for monitoring. The beam control system is called CESAR [2], which is an acronym for CERN Experimental areas Software Renovation. Through the past 10 years, CESAR has been continuously developed with new features and devices types being added. With the new secondary beams software migration project, the CESAR scope will be extended to accept optics calculations through MAD-X connectivity, and ideally also with automatic layout updates through the CERN Layout database.

The particularity of CESAR with respect to other control systems of the CERN accelerators is that it is designed to be operated by non-experts, as well. Many of the experimental users are not accelerator physicists and do not know all details of the beam line and its equipment. Therefore the system is made easy and intuitive, yet safe, in order to avoid any unintentional damage to the beam lines and experimental equipment. CESAR is based on Java and constructed around an ORACLE database. It acquires and sets so-called equipment knobs, mainly by subscribing to the Front-End Software Architecture FESA [3] device. In addition, it receives information from other services such as from the access system database (Access-DB) , via DIP (Data Interchange Protocol), and the data logging system NXCALS [4]. All devices are identified in the CESAR database together with their parameters, such as FESA name, element type, beam line, and others. This allows flexible modifications as often needed in secondary beam lines. The architecture of CESAR is shown in Fig. 1.

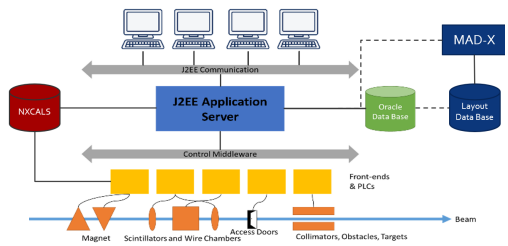


Figure 1: CESAR Architecture and Foreseen Connectivity.

USER TYPES

For both safety and simplicity reasons, there are three user types defined in the database: (1) *Super Users* are allowed to see and change any parameters of all devices in all beam lines. This group is composed of the responsible beam physicists, accelerator operators, and selected equipment specialists. (2) *Main Users* are associated with specific consoles in an experimental control room and are allowed to change most non-safety relevant settings in their beam line up to their experiment. They are set by the super users according to the experiment schedule, which is provided by the SPS/PS Physics Coordinator. (3) *Standard Users* are treated similarly as main users, however they see only their assigned experimental area, for instance to initiate an access procedure. Standard users are able to monitor their beam parameters, but are not allowed to control any devices other than the ones in their assigned user zone.

INTERFACE

The CESAR interface is composed of three main panels, as depicted in Fig. 2: the top menu, the devices panel and the beam line selection tab. The latter is used to change the

* Bastien.Rae@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

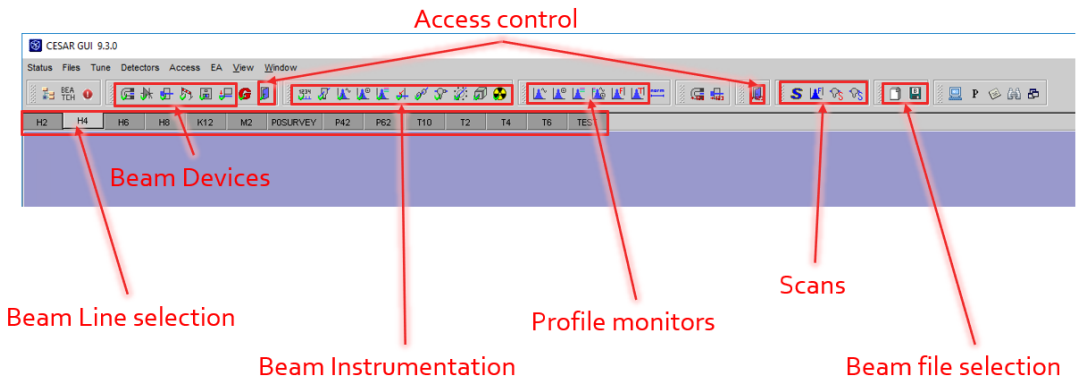


Figure 2: CESAR interface for Super Users.

selected beam line and control the devices associated with it. This functionality is reserved for super users. The devices panel displays the main features and functionality, which is needed during setting-up and tuning of beams, as well as during the operation during a physics run. On the top menu, specific panels can be opened within to the devices panel, including specific modules and panels that are related to particular beam line equipment, beam line protection, user configurations and other settings. In addition, automatic scan programs used for precise beam steering can be opened that allow efficient tuning of selected elements while visualising direct feedback by the beam instrumentation.

CESAR DEVICE CONTROL

Collimators

For collimator settings, each of the motors moving individual jaws is controlled. Collimators with four jaws are considered as two different entities, one vertical and one horizontal, for a better overview. They are used for changing intensity, shape, and energy spread of a beam. Similar to the magnet settings, one can set reference values for each of them, as can be seen in Fig. 3.

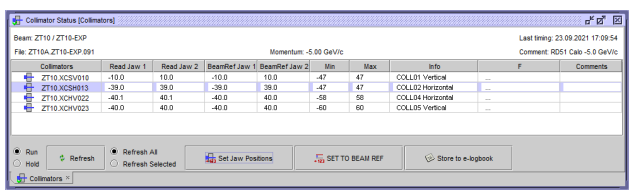


Figure 3: Collimators Status.

Magnets

In the magnets status panel (see Fig. 4), all magnets of the selected beam line are displayed together with their main parameters. There is the possibility to set and read the applied current values for each of the magnets and reference values can be defined in addition. This reference allows to go back to previous configurations, e.g. when steering the beam. CESAR also displays magnet faults together with the specific fault type, e.g. overheating. Another functionality is the so-called rectifier status, from which the power supplies can be switched on/off or moved to standby, for instance if a magnet is not in use for the currently loaded optics. It allows also resetting the power supply for certain fault types.

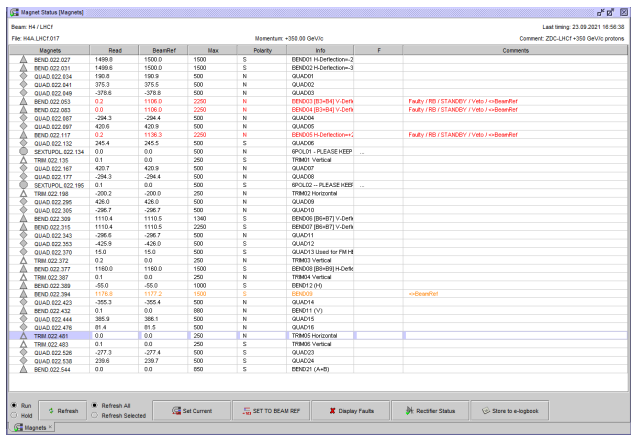


Figure 4: Magnets Status. Note the display of faults in red colour.

Obstacles

In order to create and absorb different particles types as well as for creating tertiary beams, different materials ("obstacles") can be placed in the beam. The Obstacles Command (see Fig. 5) allows the users to control the position of each device and to add or remove different kinds of material automatically as these devices are motorised. The positions are all entered in the CESAR DB, so one can directly select the desired obstacle to be placed and keep a reference, as well.

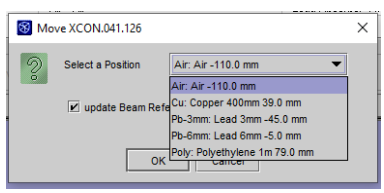


Figure 5: Obstacles Command.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV047

BEAM INSTRUMENTATION

Scintillators and Scalers

The trigger status displays counts from each scintillator along the selected beam line, as depicted in Fig. 6. In addition, it calculates 'normalised counts', which are normalised to the beam intensity on the upstream primary target in order to avoid fluctuations coming from the primary beam. As they are motorised, scintillators can be moved out of beam on demand, e.g. to reduce absorption for low-momentum electrons. Furthermore, in each control room, users can connect their discriminated NIM detector signals to scaler units, which are then displayed on CESAR and allow beam operators to scan and set the beam position for a maximum number of counts.

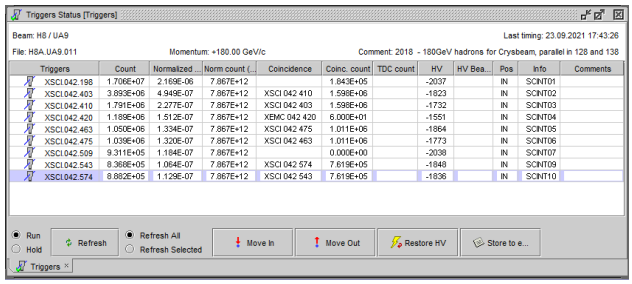


Figure 6: Scintillator Status.

Profile Monitors

As shown in Fig. 7, CESAR displays beam profiles along the beam lines independent of the type of monitors that are used. Typical monitor types are analogue MWPCs, delay wire chambers and scintillating fibre monitors (XBPF). CESAR provides count rates from each monitor as well as calculated mean values of the profile distribution. As for the scintillators, some of the monitors can be moved out of the beam. Voltage settings can be adjusted by the operators for an optimal dynamic range.

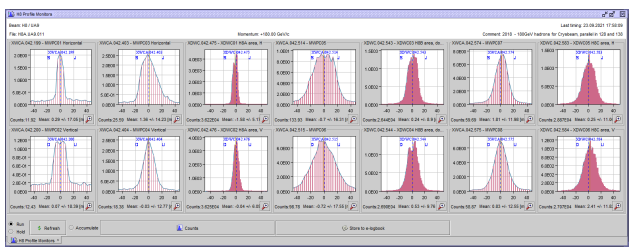


Figure 7: Profile Monitors.

Other Instrumentation

In addition to the above, CESAR offers displaying a variety of other beam instrumentation types specific to the selected beam line. Such detectors include the FISC scintilating wire scanners, for which CESAR reads counts versus the selected fibre position, and Cherenkov detectors for beam particle identification, for which users can set the gas pressure and even make pressure scans besides the readout of the counting rate.

IMPROVED OPERATIONAL FEATURES

Scans

CESAR offers the possibility to perform scans on any beam device and instrumentation. One can select the control element (e.g. magnet or collimator) and the instrumentation to perform a scan between certain values in selected steps. The scan will go through all preset values and plot the detector reading as a function of scanned parameter, e.g. a magnet current as depicted in Fig.8. This needed allows to maximise transmission through a beam line or to find the position of a user detector without the need of survey in the zone. FISC scans can be performed in different modes, i.e. one position per extraction or in a fast mode for a complete scan during one extraction. There are different expert modes in addition, for instance to scan the beam divergence between two FISC monitors.

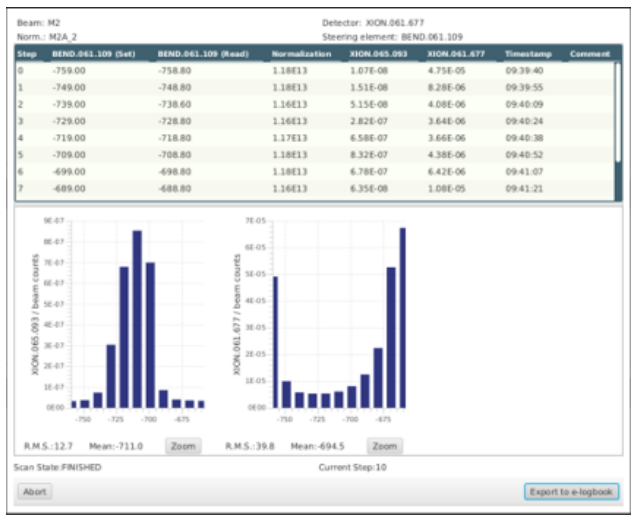


Figure 8: Scan Result.

Beam Files

Users of experimental areas may want different beam parameters and desire different particle types, energies, and intensities. The beam files of CESAR contain all beam reference values for the selected beam lines that have been set and saved after commissioning of the specific optics and beam. This allows users to switch easily between configurations as needed by simply loading a beam file without the need of constant help of the beam operators and physicists. In addition, each file can be extrapolated to different energies, taking into account energy changes in the line as needed for tertiary beams.

Access System

Users of secondary beams typically often need access to their respective experimental areas, in particular for test

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

beam users. It is therefore important to allow quick access to their setups without having to ask operators constantly. Hence, an access system control has been implemented in CESAR. All users can see the access status of their experimental area. The main user can both ask for access in their respective zone and turn on the beam for all users in the beam line. For safety reasons, CESAR receives information from the access system and commands the safety devices only if allowed and not prohibited by the beam interlocks. In order to make sure that no erroneous command can be sent to a device protected by the access system hardware loop, the access system matrix including all safety elements is duplicated in CESAR and verified before sending the signal. In this way, it is ensured that all conditions are fulfilled before opening a zone or turning on the beam.

Radiation Monitors

Reading of radiation monitors has been implemented in CESAR for each beam line, in order to check the ambient radiation levels as well as to display information about the alarm thresholds. If the warning threshold is passed, the respective line will turn orange in the display window. The colour changes to red if the alarm threshold is passed. This allows a quick follow-up in case radiation alarms occur. Note that the radiation safety system is completely independent and CESAR only displays information for convenience.

Beam Layout

The beam layout panel displays all devices that are registered in the CESAR DB for the selected beam line. In the experimental areas, the beam lines change regularly depending on user requests. If some equipment is removed from the line, the super users can hide devices that are not needed. Those devices then are not shown in the regular status panel anymore in order to keep the overview concise. In addition, super users can "protect" any device, which means that any other user is inhibited from operating it anymore. Finally, Super users can add comments to any device that will be displayed in their corresponding panel, e.g. for better explanation of the device function or to highlight important features of the device setting.

SOFTWARE MIGRATION PROJECT

The Software Migration project has been initiated with the goal of modernisation of the offline software used for the description and design of the secondary beam lines at CERN. The situation at the beginning of the project in 2017, the reasons for undertaking the migration, and the baseline software after the completion of the migration by the time of the restart of the Experimental Areas beam lines after the Long Shutdown 2 have been described in [5]. The present contribution summarises the project status and puts a particular emphasis on the coupling of the software used for the beam optics calculations (MADX and AppLE.py) with CESAR.

Project Status

A migration of the complete software chain used for the design of the secondary beam lines in both CERN North and East Areas has been performed. The new baseline consists of MADX [6] for beam optics and survey calculations, the in-house developed software AppLE.py for graphical output and matching, as well as FLUKA [7–9], BDSIM [10] and Geant4 [11], respectivel Geant4-based derivatives, for beam-matter interactions. The solution has been validated with the help of benchmark studies and a test of the complete software chain. It is planned to use the software in a highly integrated way, utilising the modern online database tools available at CERN, such as Layout Database and GitLab.

The new software has become the baseline for Run 3 (2021 - 2025), which is expected to allow the final validation of its practicality and to reveal some aspects requiring improvement. While the major migration work has been completed according to the initial project plan, the work on the adaptation of the software chain to the evolving software infrastructure at CERN as well as the integration of the beam lines into the Layout Database, the benchmarking studies and work on further automatisation are foreseen to continue during Run 3 and beyond.

Layout Database

The Layout Database [12] is a CERN-wide database, designed to contain integration and installation layout data, a naming portal, photographs and drawings of the beam lines, tunnels, areas, as well as tables with all parameters relevant for the beam line description for the CERN accelerator complex. The secondary beamlines are currently being included in the database in the framework of the CERN-wide Configuration Management. It is planned to import beam line parameters such as magnet names, magnetic lengths, apertures, mapping of magnetic field strength to currents and vice versa as well as others from the CESAR database into the Layout Database and vice versa. For the latter, there exists a function of automatic generation of MADX input files from the Layout Database, which has been adapted to match the format and naming convention as required. This application takes various parameters from the database and constructs the MADX input in form of a sequence file the given beam line. This tool has been successfully tested with the K12 beam line and the sequence file has been validated successfully with the help of the previously used software. Many of the use cases for the new software chain will be tested now thanks to the restart of beams after LS2. A large share of the North Area beam lines has still to be implemented into the Layout Database, which is planned to be completed by the end of 2021.

Envisaged Future Steps

Continuing the integration of the North Area beam lines into the Layout Data Base and the MADX sequence file generation for each of them. In the medium-term, it is envisaged to create an interface between AppLE.py and CESAR.The

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV047

first steps for such an integration have been taken already, allowing beam files from the CESAR database to be read by AppLE.py. That way the beam optics for any specific beam file can be calculated and visualised, allowing for instance to predict losses at collimator apertures. It is also planned to feed back newly generated and modified optics settings from AppLE.py to CESAR beam files

CESAR FUTURE

The most important aspect from the configuration management point-of-view will be the connection of CESAR to the newly commissioned beam software. The project is on a good track and several new features for CESAR have been already developed, such as the Apple.py-to-CESAR conversion and the automatic layout update with the Layout Data Base. We are thankful for the plenitude of ideas reaching us from the user community and from the recently established North Area Consolidation Project, which are evaluated at the moment. A frequently wished for item is establishing an Application Programming Interface (API) for CESAR, permitting Super Users to access the CESAR functionality from within scripts. This would allow to automatise even complicated steps for beam tuning with direct feedback from the beam instrumentation. In addition, connecting CESAR to the NXCALS logging service will allow users to retrieve recorded values of any device in convenient way. Thinking further ahead, integrating fault reporting into CESAR, e.g. with the already existing Automatic Fault System AFT [14], will improve reliability analyses and save time of the operators.

In addition, the new CERN GUI Strategy working group currently reviews the existing GUI systems with the aim of streamlining and easier maintainability. This is a good opportunity to improve the graphical interface and to explore possible synergies with the other control systems of CERN, for instance by adding some useful features that have been developed for accelerator controls.

CONCLUSION

CESAR is a versatile and flexible control software that is used in the experimental areas of CERN allowing users the operation of all beam devices in the secondary beam lines. It features personalised settings, such as beam configuration files, which enables quick changes of beam parameters up to a complete change of particle species and beam momentum. CESAR is being improved continuously with new features becoming available and following the evolution of users requirements. Recently, in the framework of the secondary beam software migration project, a first interface to beam simulations has been established that will allow visualisation of models, currently loaded optics and direct feedback from beam instrumentation. In the future, further upgrades are envisaged, reaching the full capabilities due to the software migration project, the North Area Consolidation Project, and the new CERN GUI Strategy.

ACKNOWLEDGEMENTS

The authors warmly thank G.L. D'Allesandro, D. Walter, I. Perez, M. van Dijk, M. Rosenthal, and E. Montbarbon for their important contributions to the software migration and the CERN management for their continuous support of these activities.

REFERENCES

[1] D. Banerjee, J. Bernhard, M. Brugger, N. Charitonidis, N. Doble, L. Gatignon, A. Gerbershagen, "The North Experimental Area at the Cern Super Proton Synchrotron," CERN-ACC-NOTE-2021-0015, 2021.

[2] V. Baggiolini, P. Bailly, B. Chauchaix, F. Follin, J. Fullerton, P. Malacarne, L. Mateos-Miret and L. Pereira, "The CESAR project: Using J2EE for accelerator controls," CERN-AB-2004-001-CO, 2004.

[3] M. Arruat, L. Fernandez, S. Jackson, F. Locci, J.-L. Nougaret, M. Peryt, A. Radeva, M. Sobczak and M.V. Eynden,"Front-End Software Architecture", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.

[4] J. P. Wozniak and C. Roderick, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper WEPHA163. doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[5] A. Gerbershagen, D. Banerjee, J. Bernhard, M. Brugger, N. Charitonidis, G. L. D'Alessandro, L. Gatignon, E. Montbarbon, I. Peres, B. Rae, M. Rosenthal, M. Van Dijk, "CERN Secondary Beamlines Software Migration Project", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOPHA047. doi:10.18429/JACoW-ICALEPCS2019-MOPHA047

[6] L. Deniau, H. Grote, G. Roy, F. Schmidt, "The MAD-X Program User's Reference Manual", CERN, 2021.

[7] G. Battistoni et.al., "Overview of the FLUKA code," Annals of Nuclear Energy 82, 10-18, 2015.

[8] T.T. Bohlen et.al., "The FLUKA Code: Developments and Challenges for High Energy and Medical Applications," Nuclear Data Sheets 120, 211-214, 2014.

[9] V. Vlachoudis, "FLAIR: A Powerful But User Friendly Graphical Interface For FLUKA", in Proc. Int. Conf. on Mathematics, Computational Methods and Reactor Physics, 2009.

[10] L.J. Nevay et al., "BDSIM: An Accelerator Tracking Code with Particle-Matter Interactions," Computer Physics Communications 252 107200, 2020.

[11] S.Agostinelli et.al., "Geant4 - a simulation toolkit," NIM A, vol. 506, no. 3, pp. 250-303, 2003.

[12] R. Billen, J. Mariethoz, and P. Le Roux, "The LHC Functional Layout Database as Foundation of the Controls System", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper RPPA03, pp. 526–528.

[13] G. Burton, M. Hanney, P. Strolin, "BEATCH: a Fortran programme for the particle optics of beam transfer channels," ISR-BT-TH/69-27, 1967.

[14] C. Roderick, L. Burdzanowski, D. Martin Anido, S. Pade and P. Wilk, "Accelerator Fault Tracking at CERN", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 397–400. doi:10.18429/JACoW-ICALEPCS2017-TUPHA013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV048

UPDATES AND REMOTE CHALLENGES FOR IBEX, BEAMLINE CONTROL AT ISIS PULSED NEUTRON AND MUON SOURCE

F. A. Akeroyd, K. V. L. Baker, L. Cole, J. R. Harper, D. Keymer, J. C. King, T. Lohnert, A. J. Long, C. Moreton-Smith, D. Oram, B. Rai, Science and Technology Facilities Council, Rutherford Appleton Laboratory, Chilton, UK

Abstract

IBEX is the EPICS based experiment control system now running on most of the beamlines at the ISIS Neutron and Muon Source, with plans to deploy to all remaining beamlines by the end of the upcoming long shutdown.

Over the last couple of years we have added support for reflectometry and muon instruments, developed a script generator, moved from Python 2 to Python 3, and continued to build on our suite of device emulators and tests. The reflectometry inclusions required the development of a framework to maintain the complex motion control requirements for that science technique.

Whilst it is desirable that IBEX is easily configurable, not all operations should be available to all users, so we have implemented functionality to manage such access.

The COVID-19 pandemic has meant we have also had to adapt to greater amounts of remote experiment access, for which we developed systems covering both IBEX and the old SECI control system.

This presentation will aim to provide a brief update on the recent changes to IBEX, as well as outlining the remote operation solutions employed.

INTRODUCTION

The ISIS pulsed neutron and muon source [1] is a world-leading centre for research in the physical and life sciences and currently has over thirty beamline instruments. The IBEX control system [2, 3] is currently replacing the previous control system, called SECI, and to date about two thirds of instruments have been converted. IBEX is a client-server based system composed of EPICS [4] for the server part, Eclipse/RCP/Control System Studio [5] for the client GUI (see Fig. 1), and utilising Python for scripting.

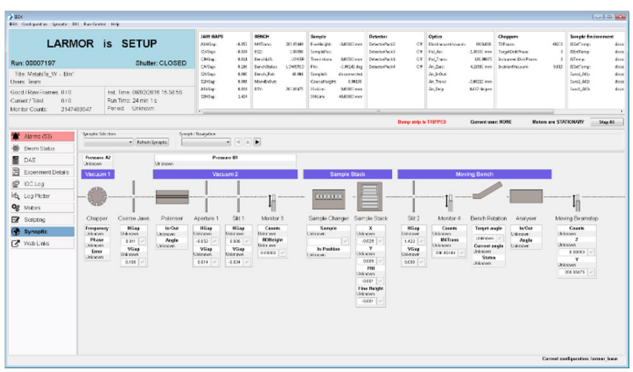


Figure 1: The IBEX user interface.

REMOTE EXPERIMENT ACCESS

Lockdowns and subsequent travel restrictions, social distancing and new safety measures brought about by the COVID-19 pandemic caused disruption to planned operation of the ISIS facility. When systems were put in place to allow the accelerator and instruments (beamlines) to operate again, it was still impossible for most external facility users to travel to ISIS to perform an experiment.

Gathering Requirements

A consultation was undertaken with instrument scientists and technicians, initially with a questionnaire and then follow up meetings with group leaders. The aim was to determine how they foresaw operations with "user not present" experiments and to look for common themes of where best to concentrate our resources. We also needed to cover both IBEX and non-IBEX (SECI) instruments with our solutions.

There were quite a range of levels of user engagement proposed, these were often determined by safety or equipment damage considerations with remote control. Only one instrument group, the muon beamlines, wished their users to have full access. Other instrument groups wished various degrees of read-only access so the user could monitor the experiment remotely, liaising with the local contact when necessary. The support technicians also wished for enhanced remote monitoring, which would allow them to better optimise timing and resourcing of general equipment tasks on site.

Full Remote Experiment Access

For the muon beamlines it was practical to allow remote users full access to the experiment equipment. We were able to achieve this by using the cloud connectivity solution provided by the commercial RealVNC package [6], effectively putting the user in the instrument cabin. The muon beamlines were not fully running IBEX at this point, so this solution also avoided having to modify the old SECI control system.

A dedicated control computer in each instrument cabin was configured for cloud VNC access, instrument scientists then added users to the relevant VNC group granting access only for the duration of their scheduled experiment. Remote users were required to use Two-Factor Authentication (2FA) [7] for access to the VNC system.

In addition the VNC system has proved useful for more than just remote user experiments. An engineer from an overseas external company was also able to take part in commissioning a new chopper system along with local staff.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV048

Though VNC was primarily used for read/write access, it can be used to grant read-only access to a computer. Such access, however, does not even let you change which window you are viewing, so is only useful if there is a particularly relevant screen that could be left visible on the system

Remote Monitoring via Web

ISIS has a "web dashboard" (see Fig. 2) for both old and new control systems, this gives a very brief snapshot of the instrument state and key values (blocks), but is more suited to quickly checking the instrument is still working or what stage an experiment has reached. To monitor progress in sufficient detail, the ability to see historical values is required.

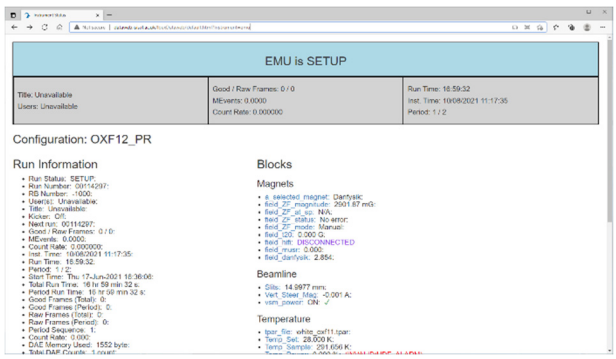


Figure 2: Instrument web dashboard.

Developing a solution applicable for both the IBEX and the SECI control system made use of a common component of both systems, the ISISICP program. This manages the fast data acquisition system, but also has access to all the sample environment data as it is responsible for writing the final NeXus HDF5 [8] experiment data file.

We chose to use Grafana [9] to provide the graphs of experiment parameters. The system is stationed behind an Apache web server proxy and uses a FreeRADIUS plugin to interact with the user office system to authenticate users and check they were part of the appropriate "virtual visit".

To provide the data for Grafana, we chose to make use of the Prometheus [10] monitoring tool as an intermediary. The common control program component was instrumented to allow it to be scraped by the Prometheus system, which then stored the values (metrics) in its time series database. A data retention period of two weeks was applied in Prometheus to the scraped data. As Prometheus is a full monitoring system it also supports defining rule-based alerts, but as we already had an existing system for this it was not something we investigated.

Figure 3: Plot of a block from the web dashboard.

A simple single value plot from following a link to a block on the web dashboard is shown in Fig. 3. A more complex interaction can be obtained via the Grafana explorer interface as shown in Fig. 4. Items monitored via Prometheus conveniently appear in the metrics dropdown list and multiple queries can be added to the same graph and then stored in the query history for use at another time.

The Grafana data view proved very popular with the various groups. The technicians also created their own cross-instrument dashboard to help with coordinating work.

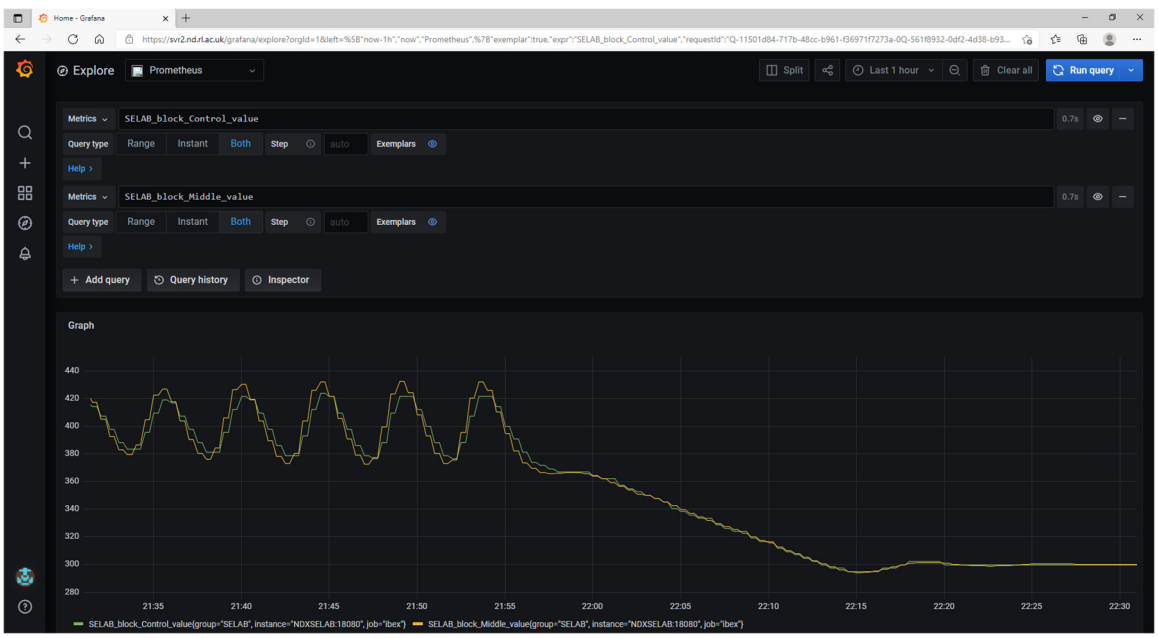


Figure 4: Grafana explorer view.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Remote Monitoring via IBEX Client GUI

Many ISIS users were already making use of IDAaaS (ISIS Data Analysis as a Service) [11] for carrying out data analysis using software such as Mantid [12]. The IDAaaS system allows authenticated users to provision a new pre-configured Linux virtual machine which has appropriate resources and pre-installed software for a particular data analysis technique.

Though the IBEX control system primarily runs on Microsoft Windows, this is only (currently) a requirement for the server part of the system. The IBEX client is Java based, making use of Control System Studio and Eclipse/RCP, so is cross platform. By providing the IBEX client on the IDAaaS system we were able to give users an enhanced read-only view of the experiment, they were able to open all equipment panels from the synoptic and plot values in the data browser window, even access the live detector view served via channel access through EPICS areaDetector [13]. Users were, however, not able to change values via this interface, this access was enforced via EPICS CA gateways.

The IBEX GUI did not require extensive work to be packaged and deployed to the linux IDAaaS system. Most issues were related to accidental use of \ rather than / in pathnames and mis-matching of upper/lower case between on disk and in software filenames. These issues were corrected and automated checks added to our system.

BUILDING AND DEPLOYING SYSTEMS

The IBEX control system runs on Windows virtual machines, which provides a good business fit with the other IT infrastructure on site. We have been looking to improve the way we deploy and upgrade the system, both the IBEX applications and the Windows operating system.

Over time the performance of an operating system can suffer from repeated installation/removal/upgrade of applications and security patches; in additional different instruments can diverge in the type and version of software installed. Ideally, we would like to refresh the operating system while preserving the rest of the system.

Our new Windows 10 instrument system is generated by using the Microsoft Deployment Toolkit (MDT) [14] to build a system image with relevant applications installed, and then to attach the remaining parts of the IBEX control system as separate Virtual Hard Disks (VHDs) that have been separately created elsewhere. Applications that make use of the windows registry are installed as part of the base Windows system image, however IBEX and most of its associated utilities are purely file based and can be deployed on the non-operating system VHDs. Besides the main Windows system VHD, we have four others that are attached to the virtual machine:

- APPS: main applications minus any instrument specific settings e.g. EPICS Input/Output Controllers (IOCs)
- SETTINGS: instrument specific details and configuration information, referenced by IOCs on APPS VHD
- SCRATCH: temporary storage, raw data cache
- VAR: MySQL database files, other dynamic settings

By using MDT we have embedded the knowledge of building and configuring the control system computer into version controlled scripts. Performing a major windows and applications upgrade would now be easier and involve generating a new windows base image, mounting an updated APPS VHD, mounting the other existing VHDs, and running the upgrade script provided on the new APPS VHD which will make any required changes to files on the SETTINGS and VAR VHDs.

REFLECTOMETRY SERVER

One of the most complicated scientific techniques from the controls perspective is reflectometry. These beamlines have many axes of motion and require precise alignment as well as coordinated motion of components. Items commonly found on other beamlines, such as slits, often have an additional degree of motion on reflectometers, and super mirrors can alter the beam direction, requiring downstream components to compensate. To handle this, we developed an additional abstraction above the EPICS motor layer called the reflectometry server [15], the GUI for this is shown in Fig. 5

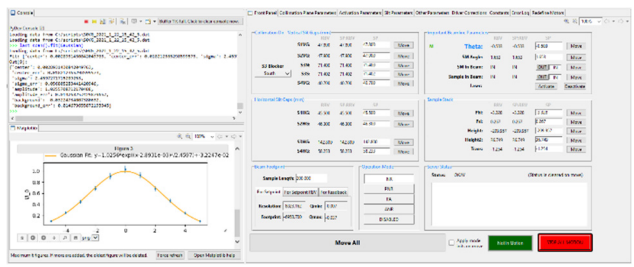


Figure 5: The reflectometry GUI.

The reflectometry server is a Python program based on the PCASpy [16] package and is configured via a Python file that defines the beamline layout. The beamline is assembled out of building blocks called components, which represent a node of interaction with the beam (either passively tracking it or actively affecting it). These components are linked to low level motors, but a management layer can provide service such as altering motor velocities to enable synchronisation of component motion and calculating a beam footprint.

The reflectometry server has recently been extended to support a bench component, which transforms three linear motion axes to motion on an arc around the sample. This has been successfully deployed to the ISIS POLREF instrument. Other improvements to the system have included:

- Allowing components multiple parking positions to choose from, these can depend on beam angle so ensuring they do not block the neutron beam
- Parking components in a sequence of multiple moves to avoid collisions
- Components can now have a variable rather than fixed Z (along beam) position, and this can be scanned.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV048

SCRIPT GENERATOR AND SERVER

Automated control of user experiments is achieved via scripts written in Python, which while powerful can be error prone. Users' scripts loaded into the system are checked with Pylint [17], which spots some errors and python 2/3 incompatibilities, but will still miss issues like referring to non–existent equipment or waiting for too much time or beam current. Other problems can occur if a user creates two scripting windows and accidentally launches two competing scripts. Running a script in a dry-run or simulation mode can spot some additional problems, but this require the script to have been written following a set of strict rules (i.e. only accessing devices via certain APIs) or else the modes will either fail or have undesired effects.

Script Generator

The IBEX script generator aims to provide a simpler table like interface for users to script the instrument. Rows are populated in a table where the columns refer to parameters that affect the control of the experiment. The parameters available and functions used to validate parameters and run scripts are configured via a local *script definition* file. More details of this system are presented in another paper at this conference [18].

Script Server

The script server is based on NICOS [19], it takes its input from the script generator and its role is to queue up scripts and provide a mechanism for interacting with the running scripts. A script server perspective is provided in the IBEX GUI; however, a future goal is to provide more dynamic interaction between the script generator and script server.

CONTROLLING USER ACCESS

The IBEX control system has several classes of user: visiting scientists, instrument scientists and technicians. Each of these groups may wish to see a slightly different view of the available information, and the ability to change specific parameters may need to be restricted by user group type.

A hierarchy of screens can be created that allows "drilling down" from a synoptic to greater details, or an "advanced" tab is sometimes used instead. Often we wish to restrict access to a screen or values on a screen, for this we have defined a system called *manager mode* that must be enabled (via entering a password in the GUI) to allow such access.

Manager mode makes use of EPICS channel access security at its heart. Process variables that require protection are placed in an EPICS access security group that only allows write access when the *manager mode* EPICS process variable is set. In addition, scripts attached to GUI panels can show/hide or enable/disable widgets based on the status of *manager mode*.

AUTOMATED TESTING

The development of IBEX utilises modern software engineering techniques, such as Continuous Integration [20], and we make extensive use of automated testing. We have developed a framework for testing our software against device emulators [21], the latter usually developed using the LeWIS package [22]. We also use Squish [23] for user interface testing.

This approach has continued to be useful to us, allowing us to spot issues and incompatibilities early when software or packages are changed.

CONCLUSIONS

The rollout of the IBEX control system to new instruments has been partly interrupted by the COVID-19 pandemic, but this also gave us the opportunity to explore and develop new remote access mechanisms that will be useful in future. The various systems and procedures that have been put in place to develop IBEX, such as extensive testing and use of emulators, taken together with the systematic creation of the instrument virtual machine system via MDT, should provide us with a good platform for future longevity and reliability.

REFERENCES

[1] The ISIS neutron and muon source, Oxfordshire, UK, https://www.isis.stfc.ac.uk/

[2] K. V. L. Baker *et al.*, "IBEX: Beamline Control at ISIS Pulsed Neutron and Muon Source", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOCPL01, pp. 59-64. doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

[3] F. A. Akeroyd *et al.*, "IBEX – an EPICS based control system for the ISIS pulsed neutron and muon source," 2018 J. Phys.: Conf. Ser. 1021 012019 doi:10.1088/1742-6596/1021/1/012019

[4] Experimental Physics and Industrial Control System (EPICS), https://epics-controls.org/

[5] Control System Studio, https://controlsystemstudio.org/

[6] RealVNC, https://www.realvnc.com/

[7] Two-Factor Authentication (TFA), https://en.wikipedia.org/wiki/Multi-factor_authentication

[8] M. Könnecke *et al.*, 2015 J. Appl. Cryst. 48 301-305 ISSN 16005767.

[9] Grafana, https:// grafana.com/

[10] Prometheus, https://prometheus.io/

[11] F. Barnsley *et al.*, "Building a prototype Data Analysis as a Service : the STFC experience.", NOBUGS 2016 Proceedings, doi: 10.17199/NOBUGS2016.65

[12] O. Arnold *et al.*, "Mantid—Data analysis and visualization package for neutron scattering and μSR experiments", Nuclear Instruments and Methods in Physics Research Section A, Volume 764, 11 November 2014, Pages 156-166, doi:10.1016/j.nima.2014.07.029

[13] EPICS areaDetector, https://github.com/areaDetector

[14] Microsoft Deployment Toolkit (MDT) https://docs.microsoft.com/en-us/windows/deployment/deploy-windows-mdt/get-started-with-the-microsoft-deployment-toolkit

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV048

[15] T. Löhnert, A. J. Long, and J. R. Holt, "Generalising the High-Level Geometry System for Reflectometry Instruments at ISIS", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA091, pp. 1300-1303. doi:10.18429/JACoW-ICALEPCS2019-WEPHA091

[16] PCASpy: Portable Channel Access Server in Python, https://pcaspy.readthedocs.io/en/latest/

[17] Pylint, https://www.pylint.org/

[18] J. C. King *et al*., "The IBEX Script Generator", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper TUPV049, this conference.

[19] NICOS, https://nicos-controls.org/

[20] Continuous Integration, https://en.wikipedia.org/wiki/Continuous_integration

[21] T. Löhnert *et al*., "Testing Tools for the IBEX Control System", presented at the 17th Int. Conf. on Accelerator and Large Experi-mental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA090, pp. 1295-1299. doi:10.18429/JACoW-ICALEPCS2019-WEPHA090

[22] LeWIS – Let's Write Intricate Simulators, https://github.com/ess-dmsc/lewis

[23] Squish, https://www.froglogic.com/squish/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV049

THE IBEX SCRIPT GENERATOR

James King, Jack Harper, Thomas Löhnert, Aaron Long, Dominic Oram,
STFC/RAL/ISIS, Chilton, Didcot, Oxon, UK

Abstract

Experiment scripting is a key element of maximising utilisation of beam time at the ISIS Neutron and Muon Source, but can be prone to typing and logic errors. The IBEX Script Generator enables collaboration between instrument scientists and users to remove the need to write a script for many experiments, so improving reliability and control.

For maximum applicability, the script generator needs to be easily configurable which is achieved by instrument scientists creating script definitions. Script definitions are Python classes that contain the parameters a user can fill in for each action in the script, and functions to execute, validate and provide a time estimation for each action.

A user organises a table of actions and fills in their values to create their experiment, these action values are validated in real time. With a valid table of actions a user can generate a Python script which can be executed via the IBEX scripting tools.

A key requirement of the script generator is for it to integrate the pre-existing Java based IBEX client. Py4J is used as a bridge between Java and the Python script definitions. An iterative, user-focused approach has been employed with quality assurance techniques such as user interface (UI) testing to achieve a behaviour-driven development workflow.

Further planned development includes dynamically controlling the execution and values of actions whilst the script is running, action iteration and user experience improvement.

INTRODUCTION

IBEX [1] is an EPICS based control system developed and used at the ISIS Neutron and Muon Source to control beamline equipment and experiments. A key feature of IBEX is the Python-based control and scripting library known as genie python [2]. This library provides functions to control experiments in an automated manner, beginning and ending data collection, writing to EPICS process variables (PVs) to control equipment amongst a host of other functionality.

Scripting is an integral part of how ISIS runs experiments, both in IBEX and the previous control system SECI. Giving users the power to write scripts allows them to control and automate experiments in a very customisable and expressive fashion. It also helps improve the reproducibility of experiments and enables maximum utilisation of beam time.

However, there are a few pitfalls when it comes to using scripting extensively. To script an experiment a user needs to understand how to code in Python, this places a steep learning curve ahead of users who have little or no experience in coding. This learning curve distracts user focus away from the science of an experiment. Furthermore, even if a user is proficient in coding they are not familiar with genie python and IBEX, so they must learn a new complex set of commands and logic in order to correctly run their experiment. For users that are familiar with the environment it is certainly easier, but writing scripts is still prone to logic errors and mistyping of commands.

There are attempts to mitigate many of these issues within genie python, such as checking of scripts for programming errors at load time, providing a reduced command set for specific instruments and providing autocomplete for PVs of interest. One of the main roles of the script generator is to mitigate the pitfalls surrounding scripting as well as enhancing the experience of a user at the facility by enabling them to focus on the experiment.

There are a number of script generators in use at ISIS with varying degrees of functionality - some that work with the previous control system SECI. The aim of the IBEX script generator, produced by the experiment controls group [3], is to provide a unified tool – taking inspiration from other script generators currently in use at ISIS – for generating scripts without having to write code for specific experiments.

WORKFLOWS AND FUNCTIONALITY

Many experiments at ISIS follow a regular pattern. Users and scientists often customise previous scripts that follow a similar pattern to the experiment they are creating. The script generator targets this workflow by allowing instrument scientists to create script definitions – which define the parameters an experiment can take and the logic to run the experiment – and users to input experiment parameters. Using these two inputs the script generator can then generate a script (see Fig. 1).

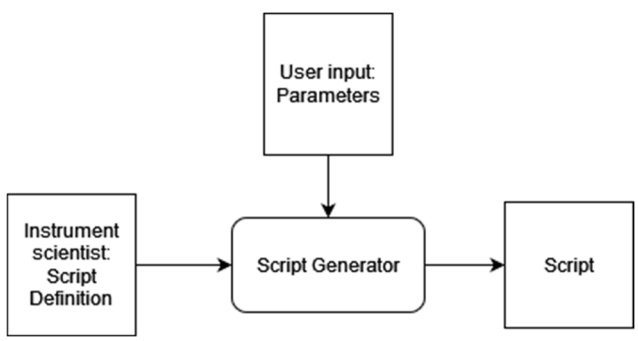


Figure 1: Script Generator inputs and outputs.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

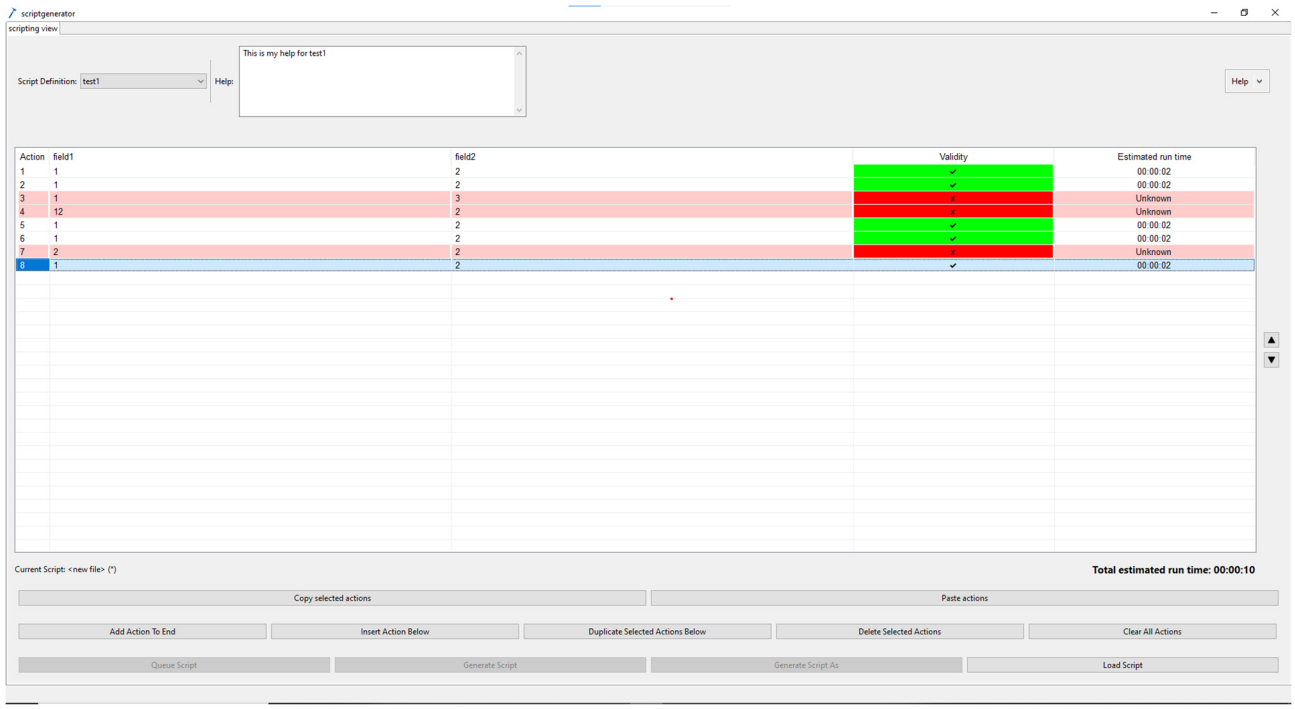


Figure 2: Script Generator User Interface.

To create their experiment the user must select the script definition (see Fig. 3) that contains the logic to run it and input their parameters into a table (see Fig. 2). In the table rows are individual actions for the script to run and columns are the experiment parameters to run each action with.

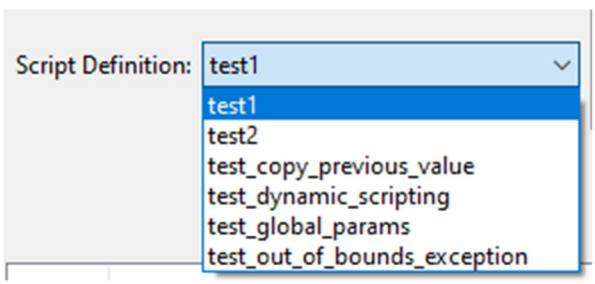


Figure 3: Script Definition selector.

There are a number of gestures (user actions) to control the order and values of actions. These include copy/paste, insert, append, delete and duplicate gestures, as well as the ability to tab between cells and reorder actions.

A user can generate a script and save it to file. This script can be executed in the IBEX scripting console. Alternatively a script can be previewed and sent to the script server [4] – a server that runs queues of Python scripts separately to the scripting console.

One requirement of the script generator is to be able to reload a script back into the script generator and edit it. To achieve this, when generating a script the parameters entered by the user are also saved to a separate json file. This file can then be reloaded into the script generator for use at a later point.

Script generator parameters are validated in real time. The validation aims to ensure incorrect values and mistyping do not end in generated scripts executing incorrectly or failing to run.

Scientists provide a validation function in a script definition. This function is arbitrary Python code which validates individual actions' parameter values. If invalid this function provides a string containing a message to display to the user as a tooltip when hovering over the action. The validity of actions are displayed through colouring an actions row.

Another feature which adds an output (read-only) column is time estimation. A script definition defines a time estimation function, which takes an actions parameter values and returns a number of seconds that this action will take to run. The time estimation column then displays this time to run in hours, minutes and seconds for each action. A total time to run is also calculated and displayed.

There are two types of parameters in the script generator, action parameters and global parameters. The previously mentioned parameters that are contained in the table are action parameters and only apply to one action. Global parameters are only set once for the whole script. For example, a script definition may use a global parameter to select a temperature controller for the whole experiment, and then individual actions may have a parameter to specify the temperature to set on that temperature controller for that action.

Global parameters are also validated individually, and can be used in the validation, time estimation and execution functions for individual actions (see Fig. 4). All these features are subject to refinement and additional functionality. This future work is discussed in the future work section.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV049

Figure 4: Global Parameters.

ARCHITECTURE

The IBEX control system uses a client-server architecture. The client provides a variety of perspectives for instrument control including the scripting console and script server. The client is developed using Java and the Eclipse RCP framework [5] and as such follows object-oriented design patterns to form the software architecture [6].

As there is a requirement to include the script generator as part of the IBEX client, it has also been written in Java using the Eclipse RCP framework.

Python was chosen as the scripting language for IBEX because it is an easy to learn language and is widely used in the sciences. Because Python is used for IBEX scripting, scientists have experience writing Python scripts, and the genie_python library has functions for instrument control, it makes sense for script definitions to also be written in Python.

Py4J [7] is a Python and Java library that enables code from each language to access objects from the other language. We are utilising Py4J as a bridge between the Java model and script definitions.

The architecture follows the Model-View-ViewModel (MVVM) architectural pattern [8] (see Fig. 5). The Java model talks via the Py4J bridge to the script definitions.

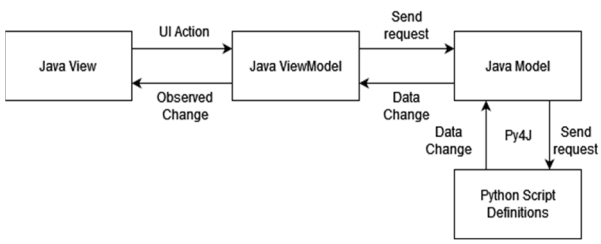


Figure 5: The Script Generator Architecture.

Python is the main language we target to generate scripts for. However, there is a requirement to be able to add new languages to generate scripts in. We have made the generation mechanism extensible using the strategy design pattern (see Fig. 7). This pattern will allow us to add extra generator languages by extending the AbstractGenerator class.

The PythonInterface class that a GeneratorPython object makes calls to is a façade on the communications over the Py4J bridge. The Python side is made up of a number of script definitions provided by instrument scientists and a number of classes providing helper methods to facilitate communication across the Py4J bridge (see Fig. 6).

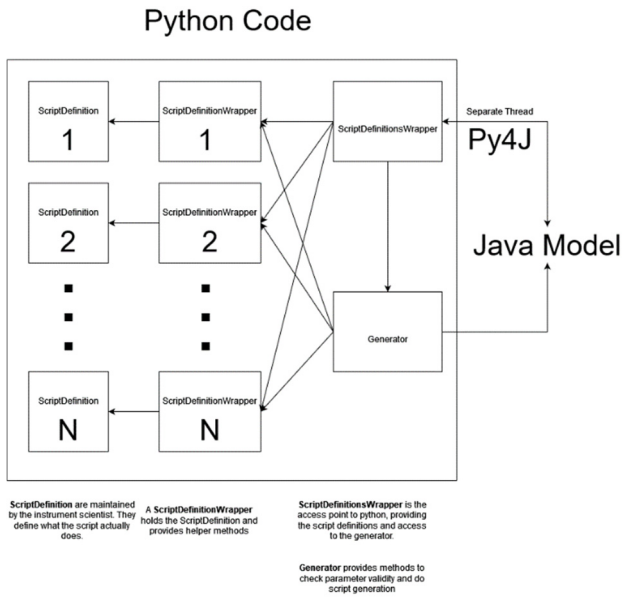


Figure 6: Python Utilities to access Script Definitions.

To compensate for the brittle and slow communication over the bridge, all communications are handled in separate threads via the Java CompletableFuture mechanism [9]. The CompletableFuture mechanism avoids blocking on the UI thread, keeping the UI responsive. However, it does mean we cannot simply set and get values from the Python side. We have thus had to develop a mechanism where the Java Model listens for the CompletableFutures to complete and passes these values back up through chain of listeners to the ViewModel from the Model.

QUALITY ASSURANCE

There are 2 forms of automated testing used to ensure the quality and behaviour of the script generator: unit testing, and system UI testing. Unit testing is carried out in Java using the JUnit framework [10] and in Python unittest [11] is used.

Both the JUnit and unittest tests are run as part of the build process, which is regularly executed on a Jenkins [12] continuous integration pipeline.

Our system UI testing is run using a separate continuous integration pipeline. These tests are written and executed with the Squish UI testing tool [13]. Squish supports a Behaviour-Driven Development (BDD) [14] approach. Tests can be laid out in the Gherkin [15] ubiquitous language. These tests reference steps that are defined as functions which carry out actions such as press buttons, fill in table cells and verify the presence and state of UI elements.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

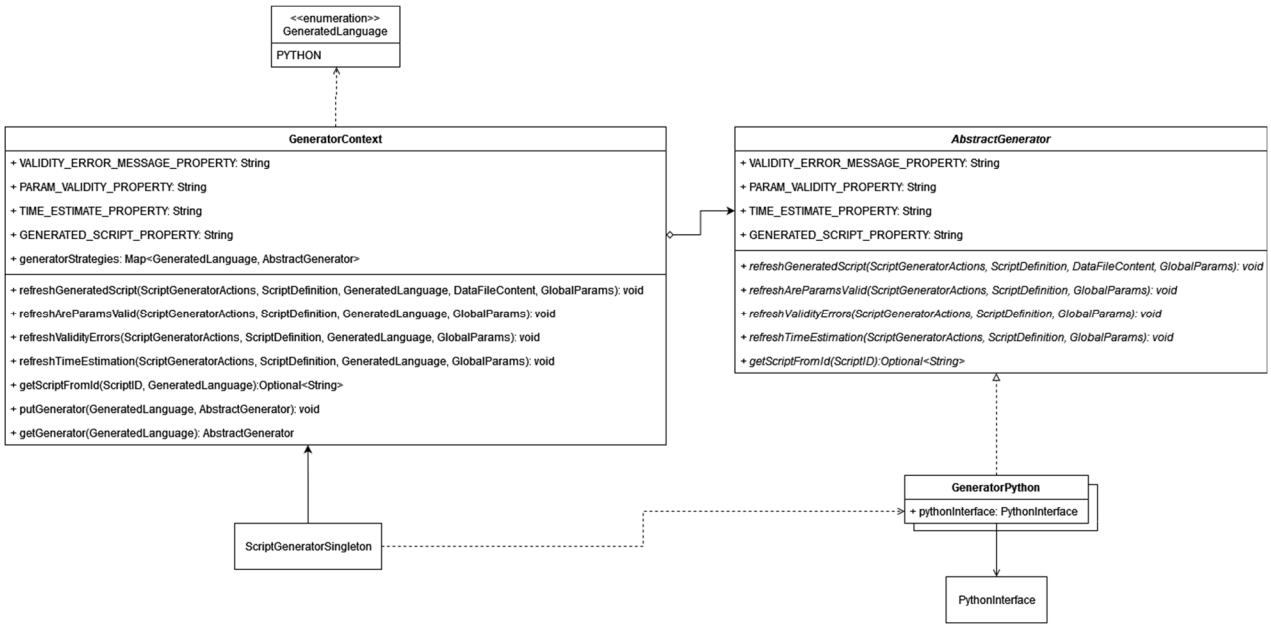


Figure 7: Strategy pattern used for generator extensibility.

We have used the Squish BDD tools [16] to describe behaviours of the script generator and verify their presence. Our suite of UI tests have caught regressions and deviations in behaviour prior to releases on multiple occasions. For example, when duplicating an action and changing the contents of one of the actions, these changes were also visible to the test on the other action revealing a shallow copy issue on the duplication mechanism.

During the release creation process we also carry out some manual system tests. These manual tests include smoke tests and tests that are difficult or impossible to automate using our current tools.

Further to testing, all code for the script generator is subject to review. These reviews help to minimise mistakes, share knowledge and techniques. Reviews also ensure consistency is practiced throughout the system, and requirements and quality standards are met. These reviews are an integral part of our quality assurance and often lead to reworks which drastically improve software quality and functionality, and the underlying code.

We run the static code analysis tool Checkstyle [17] on our Java code to identify programming flaws and enforce consistent coding standards, improving the correctness and maintainability of our codebase [18].

FUTURE OF THE SCRIPT GENERATOR

Although the script generator can be installed on any machine [19] we are current only deploying releases to two muon instruments [20] [21], for which we have met the requirements.

We are now looking at expanding the script generator functionality to make it useful to other instruments [22]. The project is being managed as a subproject of the IBEX project and fits into the scrum methodology used for the IBEX project [23]. Scrum requires work packages to be split into manageable chunks taken from requirements.

A new requirements document is being created to cover all interested parties. These requirements are being split into milestones in order to provide a roadmap to scientists and developers. The roadmap can then be used when planning work for the project.

The requirements are also split into the following categories: dynamic scripting, estimation, generated scripts, actions and parameters, gestures, script definitions, usability and miscellaneous.

Dynamic scripting captures behaviours which interact with the script server to run and control scripts directly from the script generator. Eventually a user will be able to start, stop and pause execution of the script, edit actions whilst the script is running, and execute dry runs.

Dry runs will not interact with any hardware but use a simulated version of genie python to test scripts act as expected. The script generator will display feedback from the running script to ensure the user knows the status of their running script.

Time estimation of a script is achievable with the current script generator, but scientists would also like to be able to estimate other values. For example, muon instrument scientists want to display estimated events. This is captured in the estimation category as well as displaying an estimate of when the script would finish if run at the current time.

There is some debate as to the best method of saving and reloading parameters into the script generator. Using a second file to store the parameters is confusing to users. Work to hide or remove the second file, as well as improving the readability of generated scripts and reloading of global parameters is captured by the generated scripts category.

Currently, the script generator only accepts values in the table as strings. We plan to define types for action parameters. This will enable script definitions to configure displays such as having drop downs for enumerated types and only accepting numbers as inputs for integers. This typing system is captured in the actions and parameters category.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV049

Also captured in this category is the ability to select different script definitions for each action in the table and creation of configurable objects. These objects are passed a bit like global parameters through the entire script but are decoupled from script definitions so can be used in different types of actions. Both the configurable objects and ability to select different script definitions for each action are inspired by another script generator currently in development at ISIS.

The gestures category captures the ability to export and import parameters to and from excel spreadsheets and looping more intelligently over actions.

We want to enable ease of development for script definitions to make the script generator a tool of choice for instrument scientists. Instrument scientists may not have experience in tools to manage code bases and so we have devised a workflow for managing them easily which is captured in the script definitions category.

The usability category defines work to improve the user experience, such as displaying more helpful errors when a script definition is invalid, improving the visibility of problems on the loading screen, and a UI review and subsequent redesign.

Finally, the miscellaneous category captures a few smaller work packets that do not fit into the categories listed previously, such as bugs to fix, some reorganising of where we store our script generator python code and reducing the size of the built script generator.

CONCLUSION

The IBEX script generator fulfils many of the required behaviours to improve the reliability of scripting experiments and is well on the way to being a useful tool for beamlines at the ISIS Neutron and Muon Source. However, more work is required to fulfil the requirements of those beamlines and improve the usability of the script generator.

Developing the script generator has come with a number of challenges including using a bridge between Java and Python. These challenges and the object-oriented design techniques employed have helped form the software architecture of the script generator.

We utilise several quality assurance measures to give us confidence in the functionality of the script generator. Although these measures have not caught all bugs and usability issues they have caught a number of significant problems. Our iterative approach and close working with a small number of early users have allowed us to pre-emptively identify issues and refine the script generator to improve its robustness.

Development work will continue to run within the IBEX project. A renewed focus to work with a wider user base will help us to develop a roadmap to fulfil the requirements of more beamlines at ISIS and drive future work on the script generator.

REFERENCES

[1] F.A. Akeroyd *et al.*, "IBEX - the New EPICS Based Instrument Control System at the ISIS Pulsed Neutron and Muon Source", in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, October 2015, paper MOPGF048, pp. 205-208, doi:10.18429/JACoW-ICALEPCS2015-MOPGF048

[2] Genie Python, https://github.com/ISISComputingGroup/genie_python

[3] Experiment controls group (ISIS), https://www.isis.stfc.ac.uk/Pages/Experiment-Control.aspx

[4] Nicos script server, https://nicos-controls.org/

[5] J. McAffer, J. Lemieux, and C. Aniszczyk. "Eclipse rich client platform", Addison-Wesley Professional, 2010.

[6] E. Freeman, *et al.*, "Head First Design Patterns", O'Reilly Media, Inc., 2008.

[7] M. Labanda-Jaramillo *et al.*, "Empirical Study Between Compiled, Interpreted, and Dynamic Programming Languages Applying Stable Ordering Algorithms (Case Study: Java, Python, Jython, Jpype and Py4J)", *KnE Engineering*, 2018, pp. 122-132.

[8] V. Gaudioso "Mvvm: Model-view-viewmodel", in *Foundation Expression Blend 4 with Silverlight*, Apress, 2010, pp. 341-367.

[9] CompletableFuture mechanism, https://www.baeldung.com/java-completablefuture

[10] JUnit, https://junit.org/junit5/

[11] unittest, https://docs.python.org/3/library/unittest.html

[12] Jenkins, https://www.jenkins.io/

[13] Squish, https://www.froglogic.com/squish/

[14] D. North, "Introducing bdd", *Better Software,* vol. 12, 2006.

[15] Gherkin language, https://cucumber.io/docs/gherkin/

[16] Squish BDD Tools, https://www.froglogic.com/squish/features/bdd-behavior-driven-development-testing/

[17] Checkstyle, https://checkstyle.sourceforge.io/

[18] S. Edwards, N. Kandru, and BM. Rajopal, "Investigating static analysis errors in student Java programs", in *Proc. 2017 ACM Conference on International Computing Education Research*, Tacoma, WA, USA, Aug. 2017, pp. 65-73.

[19] Installing the IBEX Script Generator, https://github.com/ISISComputingGroup/ibex_user_manual/wiki/Downloading-and-Installing-The-IBEX-Script-Generator

[20] EMU Beamline, https://www.isis.stfc.ac.uk/Pages/emu.aspx

[21] MuSR Beamline, https://www.isis.stfc.ac.uk/Pages/musr.aspx

[22] ISIS Beamlines, https://www.isis.stfc.ac.uk/Pages/Instruments.aspx

[23] K. Baker *et al.*, "Agility in Managing Experiment Control Software Systems", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper WEAR03, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV050

CONTROL SYSTEM UPGRADE OF THE HIGH-PRESSURE CELL FOR PRESSURE-JUMP X-RAY DIFFRACTION

R. Mercado*, N. Griffin, S. Lay, P. Holloway and P. Roberts
Diamond Light Source, Oxfordshire, UK

Abstract

This paper reports on the upgrade of the control system of a sample environment used to pressurise samples to 500 MPa at temperatures between -20°C and 120°C. The equipment can achieve millisecond pressure jumps for use in X-ray scattering experiments. It has been routinely available in beamline I22 at Diamond. The millisecond pressure-jump capability is unique. Example applications were the demonstration of pressure-induced formation of super crystals from PEGylated gold nano-particles and the study of controlled assembly and disassembly of nano-scale protein cages.

The project goal was to migrate the control system for improved integration to EPICS and the GDA data acquisition software. The original control system was based on National Instruments hardware controlled from LabView. The project looked at mapping the old control system hardware to alternatives in use at Diamond and migrating the control software. The paper discusses the choice of equipment used for ADC acquisition and equipment protection, using Omron PLCs and Beckhoff EtherCAT modules, a custom jump-trigger circuit, the calibration of the system and the next steps for testing the system.

INTRODUCTION

Beamline I22 and Imperial College London built and developed a sample environment [1] used to pressurise samples for diffraction experiments. This end-station equipment has been available for several years [2].

This end-station was built using a control system based on Compact DAQ measurement and control modules (National Instruments). Its associated control was realised by the collaborators using LabView.

An initial attempt was made to integrate with EPICS but the integration was not maintained, due to LabView not being an actively supported platform for the controls group at Diamond. User feedback indicated that direct control from GDA and EPICS would make operations less prone to user error.

SYSTEM DESCRIPTION

The high pressure cell has these components (as shown in Fig. 1):

- Pressure generator. This is a motor driven high pressure pump which can generate pressures up to 700 MPa. The drive motor is a high power DC servo operated with a custom built motor controller.

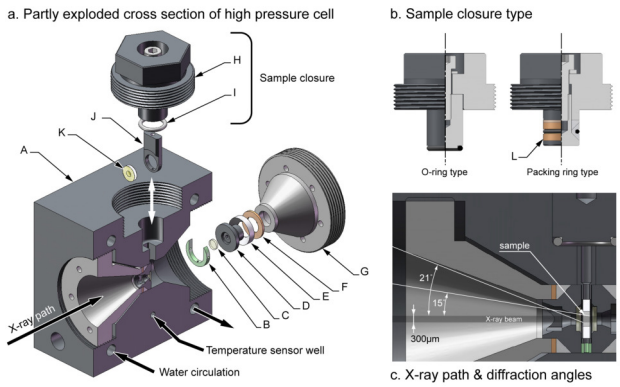


Figure 1: Detail of the pressure cell [1].

- Two remotely operated, normally closed pressure valves.
- Two remotely operated, normally open fast pressure jump valves.
- Three pressure transducers monitoring three sections of the pressure network. The sensors are 700 MPa strain gauge based transducers excited with 10 V and reading 10 mV at full scale.

The pressure network comprises three sections:

- the first section with the pressure generator,
- the second that includes an optional additional tank that can be used particularly for pressure jumps and
- the third section where the pressure cell is installed.

The control system opens and closes valves, allows the syringe pump motor to pressurise and de-pressurise the pressure network.

The pressure network has to be protected from overpressure. Valve operations are inhibited if the pressure differential is larger than 5 MPa (50 bar) between sections, with the exception of pressure jumps.

There are two jump valves, one dedicated to jumps up in pressure and another for jumps down in pressure.

HARDWARE USED IN THE UPGRADE

Diamond standardises on electronics hardware (as shown in Table 1) where possible [3, 4]. A benefit to the facility is the maintainability and support by standardising on a limited set of components.

The majority of the digital I/O signals for the system can be mapped one to one as they are standard and are based

* ronaldo.mercado@diamond.ac.uk

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV050

Table 1: Hardware Module Types and Replacements

Function	Original module	Replacement
Cell temperature PT100 sensor	RTD Input NI-9217	Omron CJ1W-TS562
Pump piston positioner linear transducer	Analogue input NI-9201	Omron CJ1W-MAD42
Motor speed control	10V Analogue output NI-9263	Omron CJ1W-MAD42
Motor ON	24V Digital Output NI-9263	Omron CJ1W-OD212
Motor Direction	24V Digital Output NI-9263	CJ1W-OD212
Valves open/close (Solenoids)	24V Digital Output NI-9263	CJ1W-OC201
Limit switches	24V Digital Input NI-9241	Omron CJ1W-ID211
Trigger for pressure jump	High speed digital IO NI-9401	Custom jump circuit
Pressure signals	Bridge input NI-9237	Strain gauge amplifier RDP DR7DC Omron CJ1W-MAD42 (Equipment protection ADC) Beckhoff EL3702 (Fast ADC capture)

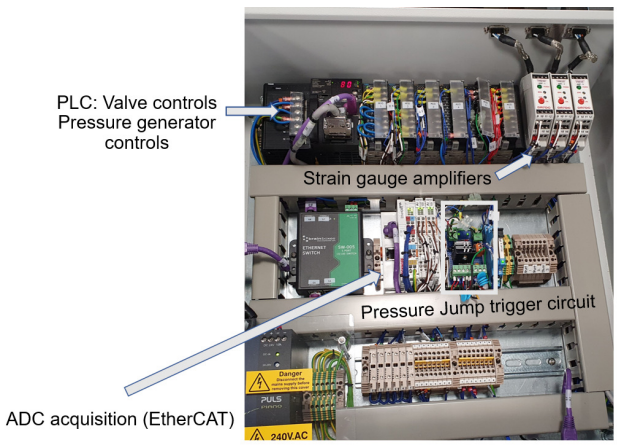


Figure 2: Electronics cabinet of the upgraded control system.

on 24 V on/off, a PT100 temperature sensor and analogue 0-10 V I/O.

Reading the strain gauge is a more specialised operation that originally used the NI-9237 module. A strain gauge amplifier was selected based on prior experience: the RDP Electronics (Wolverhampton, UK) DR7DC. The amplifier converts the millivolt strain gauge signal to the usual control range of 0-10 V. The signal is read by ADCs in parallel.

The most recent standard set of components (as shown in Fig. 2) includes a strain gauge EtherCAT module (Beckhoff EL3356), but its acquisition speed was not adequate to capture a pressure jump.

The PLC system controls the system protection, arms the pressure jumps and controls the motor for the pressure generation. The supervisory control uses EPICS. The fast EtherCAT ADC signals are captured using the ADEthercat area detector plug-in described below.

One important requirement was the desire to acquire the pressure signal during the pressure jump. An in-house electronics module was created to trigger a pressure jump on a hardware trigger signal from the beamline acquisition system. This replaced the NI-9401 high speed digital I/O module. The design was based around an inexpensive flip flop and current driver that opens a valve for a pressure jump using a TTL signal in order to synchronise with the image acquisition with the x-ray detectors. For a pressure jump acquisition the valve is "armed" so that it opens on the trigger.

During early testing the simplistic electronic design turned out to be vulnerable to noise when operating real solenoids. It took careful diagnosing and re-work of the circuit to improve the pressure jump circuit noise immunity by adding filters.

Other difficulty was signal noise from the transducers. The solution was to modify the cabling between the transducers and the strain gauge amplifiers with improved shielding and grounding.

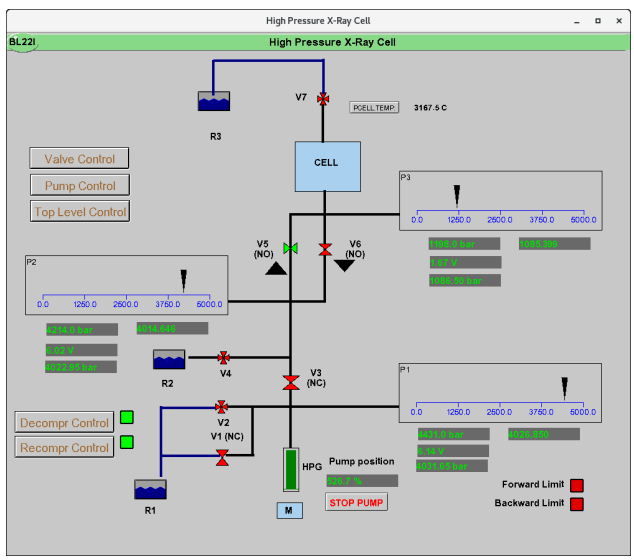


Figure 3: Top level EPICS control screen showing the pressure network.

ADETHERCAT

The pressure signals are captured at 10 kHz using the DLS EtherCAT EPICS scanner (as shown in Fig. 3). The

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV050

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ADEthercat area detector plug-in was used to capture the signal. ADEtherCAT is a variety of driver to read ADC to area detector. It was inspired by prior experience with the D-TACQ ADC (Glasgow, UK) and beam position monitor devices (e.g. quadEM [5]) that use the EPICS area detector framework to operate on 1-D signals. ADEthercat converts ADC signals from the EtherCAT master used in the Diamond ethercat EPICS driver [6] to 'image' frames. The two dimensions of the 'image' are the number of signals and 'n' samples. For the pressure cell application there are four signals captured, three for the pressures in the network and a fourth signal of the trigger value.

An immediate benefit was the ability to use a software plug-in 'NDReframe' that allows triggering on a 1-D signal as it is streamed. This solves a limitation of EtherCAT hardware that does not otherwise provide a method of triggering for data capture. The framework also adds for free the ability to write the signal to HDF5 files, which are the standard for data acquisition.

A custom pressure scaling plug-in was developed that applies the calibration method described below (Fig. 4).

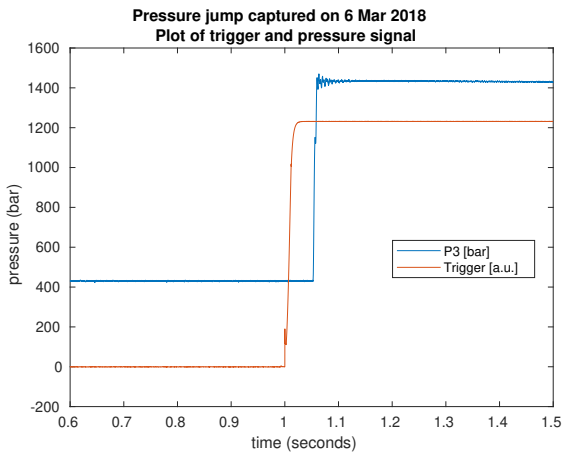


Figure 4: Plot of example pressure jump. Optimized fast valve achieves jump time of approx. 5 ms. The traces shown are the trigger (red) and pressure measurement (blue). The data capture uses the ADEthercat driver. The data scaled using a custom scaling plug-in and written to an HDF5 file using the area detector file writer plug-in.

AUTOMATED OPERATIONS

There is a mode of operation implemented at the PLC level to achieve a target pressure or a target position. It was implemented at the PLC level because it allows closer control at shorter cycle times than the supervisory control in EPICS.

There are modes of operation of higher abstraction that are implemented using a python soft IOC [7]. The operations allow setting up a pressure and setting a pressure jump.

The volume and stroke of the pressure generator are not large enough to achieve an arbitrary pressure in a single operation. Two examples are that to reach certain higher pressures the network needs to be refilled and to reach lower pressures sometimes water needs to be removed from the pressure network.

The logic allows the composition of sequences of operations to liberate the user from potentially tedious operations of pressurising the network to reach high pressures. This is more important when using the expansion tank option in the middle section of the pressure network. In tests a filling in operation needed to follow the steps to fill in and pressurise in steps for about 10 minutes with several dozen operations.

CALIBRATION

A significant hurdle to realising the project was also the calibration of the pressure signals. Calibration facilities use a calibration rig where a special oil is pressurised with known weights to reach high pressures. It was not practical or possible to operate the equipment with its full upgraded electronics read-out next to the calibrator.

Instead of this, the beamline obtained calibration certificates from ASCO Instruments (Châteaufort, France). The calibration points were linearised based on the eleven data pairs of pressure vs strain gauge voltage obtained at the calibration facility. To calibrate the electronics, the millivolt signals were input to the electronics components using a precision millivolt source (Time Electronics 1006 DC Millivolt Source Model 404S).

At lower pressures, below 50 MPa (500 bar), the calibration was changed to reflect that when opening the valves to atmospheric pressure the linearised calibration did not read zero as expected.

The result is a custom scaling mode where there are two linearisations above and below the 50 MPa threshold. This was reflected in a custom area detector pressure scaling plug-in prepared for the pressure cell and also in the PLC software.

The use of ADCs reading the amplifier signal in parallel for fast ADC capture and machine protection resulted in different voltage readings by a few millivolts after amplification. A calibration correction was adopted to account for this discrepancy that was attributed to impedance mismatch of the ADCs [8, 9].

CURRENT STATUS AND IDEAS FOR IMPROVEMENT

The system is being tested in an off-line laboratory and integration with the beamline will follow.

The project was conceived nearly five years ago. More recently it has become possible for Omron PLCs to also support EtherCAT modules that can capture ADC signals at 100 kHz. Using a higher speed CPU and the NX range of PLCs it should be possible to get the required ADC acquisition speed. The missing component is a software layer to 'stream' the ADC waveforms to area detector. This would be a welcome improvement over the current state in which limited developer effort is available to support adding more module types using the Diamond ethercat EPICS driver.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-TUPV050

The use of a single ADC instead of two sets of ADCs reading in parallel would also simplify the calibration.

ACKNOWLEDGEMENTS

Thanks to Nick Brooks for helpful guidance and discussion. Also thanks to Andy Smith for the support setting up the equipment and for obtaining the sensor calibration. We would like to thank colleagues who have contributed to the project in the past and have since left Diamond: John Emmins, Chris Roper (Engineering) and Brian Willis (Detector Group).

REFERENCES

[1] N. Brooks *et al.*, "Automated high pressure cell for pressure jump x-ray diffraction", *Review of Scientific Instruments*, vol. 81, p. 064103, 2010. https://doi.org/10.1063/1.3449332

[2] A.J.Smith *et al.*, "I22: SAXS/WAXS beamline at Diamond Light Source - an overview of 10 years operation", *Journal of Synchrotron Radiation*, vol. 28, pp. 939–947, 2021. https://doi.org/10.1107/S1600577521002113

[3] R. Mercado *et al.*, "Integrating Ethercat based IO into EPICS at Diamond", in *Proc. ICALEPCS 2011*, Grenoble, France, 2011, paper WEMAU004, pp. 662–665.

[4] M. Heron *et al.*, "Evolution of the control system standards on the Diamond Synchrotron Light Source", in *Proc. ICALEPCS 2013*, San Francisco, CA, USA, MOPPC116, pp. 381–384, 2013.

[5] Mark Rivers, quadEM EPICS module, https://cars.uchicago.edu/software/epics/quadEM.html Accessed October 2021

[6] DLS ethercat EPICS module, https://github.com/dls-controls/ethercat Accessed October 2021

[7] pythonSoftIOC module, https://dls-controls.github.io/pythonSoftIOC Accessed October 2021

[8] R. Mercado, "High pressure x-ray cell specification of pressure calculation from ADC readings", Diamond document BLS-I22-CTRL-0016, 9 March 2021.

[9] R. Mercado, "High pressure x-ray cell for soft condensed matter calibration reference", Diamond document BLS-I22-CTRL-0018, 10 May 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

IMAGE PROCESSING ALIGNMENT ALGORITHMS FOR THE OPTICAL THOMSON SCATTERING LASER AT THE NATIONAL IGNITION FACILITY

Abdul A. S. Awwal, Richard R. Leach, Jr., Roger Lowe-Webb, Karl Wilhelmsen, Vicki Miller Kamm, Bela Patel, Siddharth Patankar, Tracy Budge
Integrated Computer Control System, National Ignition Facility, Computational Engineering Division, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA

Abstract

Characterizing plasmas generated in the world's largest and most energetic laser facility, the National Ignition Facility (NIF), is an important capability for experimentalists working to achieve fusion ignition in a laboratory setting. The optical Thompson scattering (OTS) laser has been developed to understand the target implosion physics, especially for under-dense plasma conditions. A 5w probe beams can be set up for diagnosing various plasma densities. Just as the NIF laser with 192 laser beams are precisely aligned, the OTS system also requires precision alignment using a series of automated closed loop control steps. CCD images from the OTS laser (OTSL) beams are analyzed using a suite of image processing algorithms. The algorithms provide beam position measurements that are used to control motorized mirrors that steer beams to their defined desired location. In this paper, several alignment algorithms will be discussed with details on how they utilize various types of fiducials such as diffraction rings, contrasting squares and circles, octagons and very faint 5w laser beams.

INTRODUCTION

The National Ignition Facility (NIF) has made tremendous progress in our understanding of inertial confinement fusion [1]. Recent breakthroughs in fusion performance have attracted worldwide attention to NIF. In NIF, 192 precisely aligned laser beams [2] are designed to irradiate a mm scale fusion target to achieve ignition and produce a net energy gain in a laboratory setting. To achieve this goal, NIF employs several tools to diagnose the plasma condition. The optical Thomson scattering (OTS) laser [3] is such a tool which can deliver a 1 Joule, 5ω (211 nm) probe beam with a 1 ns flat-in-time pulse-shape, for diagnosing the plasma temperature at a user-specified time during the plasma evolution in a NIF experiment.

Experiments on NIF can be designed to utilize all 192 or any subset of laser beams. In every case, accurate beam alignment is essential. Beam positions are monitored in the NIF facility using hundreds of CCD cameras distributed throughout each of the 192 beams path. Feedback control loops form the main engine for the NIF automated alignment control system [2]. Position estimates are generated by image analysis algorithms, which are used to align motorized mirror pairs and other optics to direct NIF beams to their desired locations. Starting from the beam source, or Master oscillator, these alignment loops are responsible for aligning the beams along the complete beam path and finally to the mm size targets located in the target chamber center. In this paper, we will discuss several alignment loops in the OTS laser.

OTSL ALIGNMENT

A total of eight mirror pairs are closed-loop controlled using more than 14 distinct alignment loops to complete the full alignment of the OTS laser. These alignment points are divided into five main sections as shown in Fig. 1. Starting from the pre-amplifier module (PAM) and through the intermediate amplifier, the beam is taken to the Final Optics Assembly (FOA) via an imaging transport telescope section. The FOA contains frequency conversion crystals to convert the beam to the fifth harmonic and motorized, 5ω high-reflecting mirrors to point it to the diagnostic load package (DLP) located in the NIF Target Chamber. Obtaining good OTS data requires precisely overlapping the plasma volume illuminated by the fifth-harmonic probe laser and the plasma-volume imaged by the DLP OTS spectrometer.

The transport vacuum relay telescope (VRT) is used to deliver the conjugate-image of the regen apodization-aperture to the plane of the fifth-harmonic generating crystal; the associated alignment mirror pair ensures the fundamental beam is correctly pointed and centered to the FOA converter line-of-sight for optimum harmonic conversion. Two of the alignment loops discussed here align the ISP laser to the transport telescope input aperture (ISP_TTI_PL loop) and the second one aligns the output of the transport telescope to the input of the FOA (OTS_TTO_FOA_PL). Alignment images for both loops are acquired with the same far-field (FF) alignment camera in the FOA but alignment is completed for each loop using different reference fiducials. Specifically, the physical alignment reference for the OTS_ISP_TTI_PL loop is a pair of precision tooling spheres (pinheads) installed in the output optical aperture of the VRT whereas the OTS_TTO_FOA_PL loop is referenced to a commissioned pixel on the FOA FF alignment camera.

2-D and 3-D images for the OTS_TTO_FOA_PL pointing loop are shown in Figure 2. The beam has a dragonfly-shaped appearance with a distinct head and a tail. The coma-like aberrations seen in this image is caused by diffraction since the alignment beam over-fills the aperture of the intermediate amplifier rod and clips on its edge. The alignment feature for this image is the center of the head of the dragonfly-shaped beam. It was determined that a

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

centroid based algorithm [4] position detection scheme can accurately detect this feature, enabling the VRT output mirrors to be repeatably aligned. The higher brightness of the head compared to the tail, allows a high-threshold background subtraction-based centroiding, to detect the center of the head of the dragonfly.

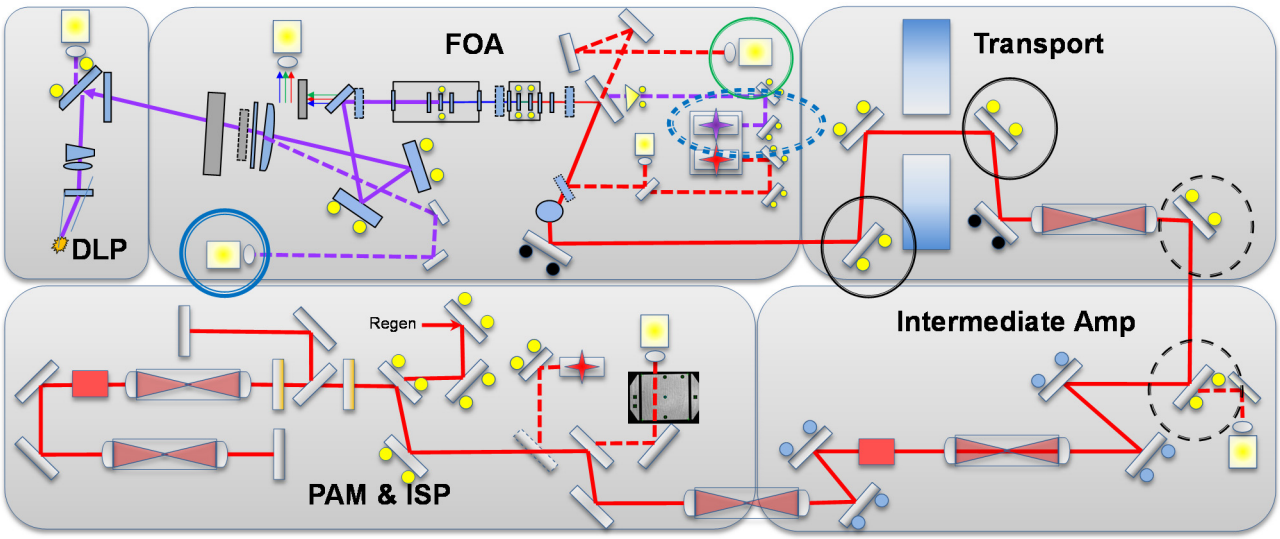


Figure 1: Alignment loops within the OTSL cover 5 sections starting from the Pre-amplifier module (PAM) and Input sensor package (ISP) through the final optics assembly (FOA) which converts the beam to the fifth harmonic and points it to the diagnostic load package (DLP). The mirror-pairs used for the OTS_ISP_TTI_PL are circled with dashed black rings whereas the those used for the OTS_TTO_FOA_PL are solid black rings. Both loops align to the same camera in the FOA circled with a green ring.

Performance of the centroid approach was tested for more than a year and was found to be satisfactory [4]. However, if there is a beam deflection with a magnitude similar to the central spot and period on the order of the image integration time, then the consequent smearing makes the centroid-approach unstable. Consider the cases of motion blur in image 59 and 229 in figure 2, where the position is shifted as the peak is shifted. Comparing the two consecutive images 58 and 59, image 58 shows a single peak with maximum at the head location. The 3-D plot of mage 59 shows lowering of the intensity of the head and elevation of the first side lobe of the tail indicative of image smearing. Other images exhibit smearing and displacement in the vertical direction, where multiple heads as well as two tails are visible, as shown in image 91 of Fig. 3.

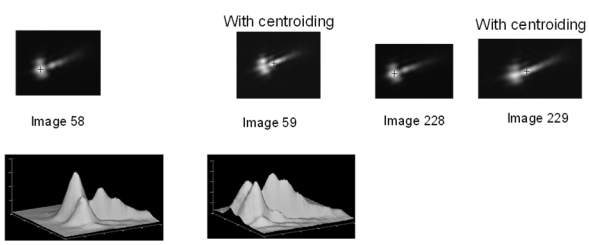


Figure 2: A typical OTSL camera image of OTS_TTO_FOA_PL loop, where OTS is the OTS laser, TTO indicates transport telescope output, (FOA --final optics assembly. PL -- pointing loop). Smearing of images causes a 19 pixel shift in centroid. The surface plot of images 58 and 59 shows the shift of the location of the maximum from the head to the tail.

This weakness in the centroid algorithm prompts us to develop an alternate algorithm.

Alternate Algorithm to Mitigate Smearing

Matched filtering is well known for its ability to track an object in a complex scene. The dragonfly shape of the OTSL pointing beam makes it a good candidate for using pattern recognition tools based on matched filtering. NIF uses matched filtering techniques [5] to provide robustness and position stability in the presence of illumination gradients, time varying signals, and unpredictable background noise. In comparison, centroid based techniques are affected by all these variables.

To test this approach, we randomly chose a nominal image as a template. Applying the same template to images 59 and 229 resulted in consistent position estimates as shown in Fig. 3. The centroid technique does not produce such a result, as observed above.

Figure 3: The result of matched filtering on the smeared image shows a sensible result.

Template Selection

When a beam is aligned, the beam may appear initially in any location within the image. To test possible variations, beam image was the shifted from the nominally aligned

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

position in four directions (left, right, up and down). 100 images were captured in each position. For comparison purposes, we apply the weighted-centroid approach to the sets of images in the nominally aligned position. The result is expressed in the scatter plot as shown on the left of Fig. 4. Next, we applied the matched filtering algorithm, using image 65 as the template. to the same set and the corresponding result is shown on the right side of the Fig. 4.

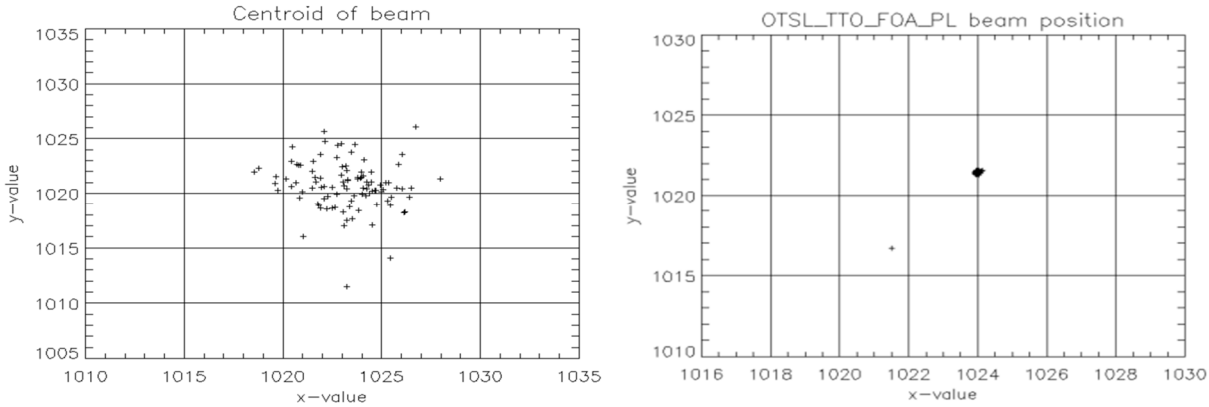


Figure 4: For a 100-image-set of an aligned beam, the scatter plot on the left is generated by the centroid approach where 99% of the spots are within a 7-pixel radius. The scatter plot on the right is generated with a template (image65). The 99% encompassing radius is reduced to a tight cluster with a span of 0.3 pixels.

Variations in intensity within the head of the dragonfly image, from image to image, produces variations in the centroid based beam position estimate.

Moving the position of the beam can cause the shape to change. However, in practice, the measurement will be for a consistent spot when it is in the fixed aligned position, for that reason the align65 template was deployed. Next, we look at the second OTSL alignment loop.

ALIGNING DIFFRACTION RING

Alignment fiducials in the beam image are often used to help locate the beam or reference position. One of the OTSL alignment loops, ISP_TTI_PL, uses out of focus imaging of a spherical object connected to a shaft as a fiducial. These fiducials are positioned at the edge of the optical aperture of the circularly formatted alignment laser beam. The alignment camera is focused to image the NF relay plane which is about 40 meters away from the pointing reference fiducials. The beam leading to a clipped diffraction pattern The out-of-focus imaging produces a pair of diffraction rings at the top and bottom of the alignment image as shown in Figure 5. Both rings are present with partially missing portion. The objective of this alignment algorithm is to detect the center of the diffraction rings under various imaging conditions.

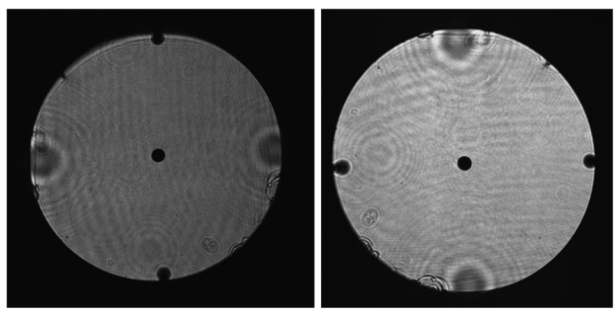


Figure 5: (a) A typical OTSL camera image ISP_TTI_PL loop, where ISP is the ISP laser, TTI indicates transport telescope input, and PL stands for pointing loop. Every beam is aligned using a pointing and centering move (b) The diffraction rings may appear in the top and bottom, note that the left ring was moved to the top.

Algorithm

The diffraction rings intensity levels are very similar to the background making a threshold-based approach centroid approach not possible for meeting alignment requirements. Since the diffraction ring has a distinct pattern however, pattern recognition techniques based on matched filtering [5] is a viable option. As a first approximation, the template is considered to be a circular ring, with a fraction missing to resemble the missing part of the diffraction as shown in Fig. 6. The parameters inner radius, r_est_left, ring thickness, delta, percentage of the ring missing from one side, missing fraction defines the template. If missing fraction is 50% then half of the ring is missing.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

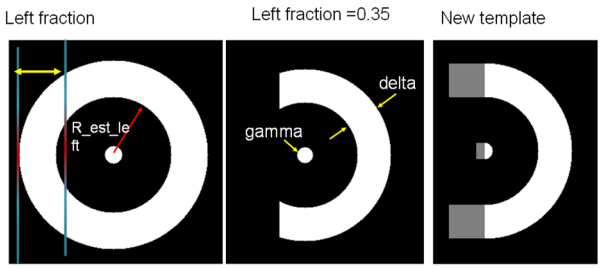


Figure 6: The template showing 4 parameters defining it: estimated radius for left (R_est_left), radius of inner spot (gamma), width of the ring (delta) and missing fraction (left fraction). The new template design instead of a full circular ring use half circle with linear extension (on the right).

To avoid spurious correlation peaks, two rings are initially segmented using the angular location from a global centroid [6]. During alignment, the missing fraction for both rings changes. It was found that if the missing fraction of both rings remained same, the template in Fig. 6 performed well, however, as the image moved in four directions the template was unable to detect the center accurately. To correct an incorrect position using angular position and distance, one must be certain which of the two position is correct.

Certainty Detector

As inspired by a bipolar binary associative memory [7] single computational neuron, is used to evaluate the quality of the location detected. The function of the neuron is to generate a high score when there is a match of the stored pattern with the input pattern and a low score when there is no match. Here the stored pattern is the cross-section of the template in bipolar binary. When the position is correct, high score of above 80 is obtained. For a totally incorrect position, the score is highly negative. These scores are used to generate the uncertainty of the position as well as correct an incorrect position.

Template Design II

A simulation of the Nfiducial diffraction pattern reveals that the ideal diffraction pattern may not actually contain a complete circular ring, rather it is partially circular and partly linear. Therefore, the template was redesigned as depicted in the right side of Fig. 6, where a half circular ring is extended by the straight edge of the template in a linear fashion up to an extent defined by the missing fraction. The weight of the linear extension was 50% compared to the weight of the right half, to reduce the effect of the high intensity near the edges which has biased the previous centering position.

Optimization

During alignment, the rings do move relative to the large disc, and as such the rings appear with different missing fraction as shown in Fig. 7. To capture the possible variations of the alignment images, the nominally aligned position image (used in Phase I) was moved in four directions to the left, right, up, and down and 100 images were captured in each position. One image from the set is shown in Fig. 7.

To choose the optimized parameters defining the template, the position detection program is executed in a loop with varying four parameters: inner radius, width of the ring, radius of the inner disc, and missing fraction on single image for both left and right rings in the aligned set of 100 images. Solution which gives minimum divergence of each position in terms of standard deviation is chosen after visual verification. Then process of stability check is then applied to the remaining 400 image sets at all the other positions above. After that the solution is applied to a data set taken from the images archived in the NIF for the previous 6 months. Using this data set, a variety of templates was applied until a solution was found that worked robustly on the entire set of NIF images.

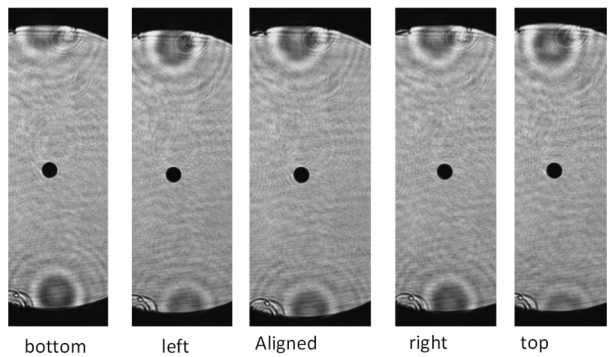


Figure 7: The variation of the diffraction rings as it moves left right and up and down with respect to the aligned position.

Figure 8 shows the difference between templates I and II design. The left results show, the phase I template introduce bias in the lower position. In phase II, the result is improved significantly with lower incidence of bias towards any specific direction as shown in the right onset.

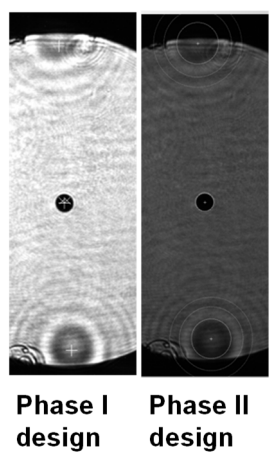


Figure 8: The variation of the diffraction rings as it moves left right and up and down with respect to the aligned position.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The optimized parameters are, top template radius = 77, width = 48, missing fraction = 0.3 and gamma = 12 and the bottom template radius = 73 width = 45, missing fraction = 0.6 and gamma = 12. After position estimation, the missing fraction is re-calculated. If the supplied missing fraction was very different from the re-estimation of the same, and it also had a negative score, only then was the algorithm allowed to recalculate the predicted second spot. Most of the time, the new missing fraction calculated online was used to stop predicting the second spot and avoid oscillation, which could make the alignment in a closed loop very difficult. Correlation-based approaches must be guarded against spurious detection. After the two rings are located, the position data obtained is subjected to several spacing tests. Such spacing tests with certain tolerance help us to eliminate spurious spots from the solution.

Next, we look at an alignment loop algorithm for centering the OTSL 5w laser.

OTSL SQUARE BEAM ALIGNMENT

One of the more challenging image processing loops in OTSL is the AA_OTS_PILOT_CL loop. The image processing algorithm for this loop uses multiple, superimposed image features to locate the beam reference and the beam center to enable proper laser alignment. The features consist of parallel opposing beam edges (reference location) and parallel opposing lines or dark bands (beam location) to estimate the 5w beam centering offsets. The images for this loop tend to have high noise within the image as well as near desired beam edge features. In addition, image illumination is often poor and can have undesired gradients near the beam edges. A typical image example is shown in Fig. 9.

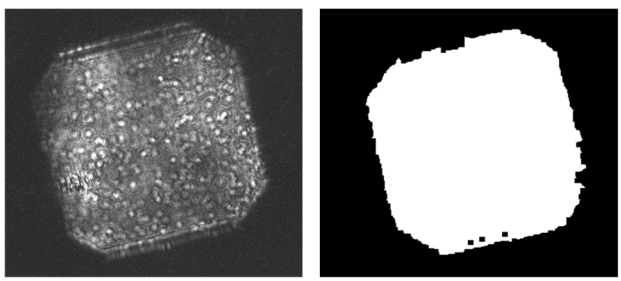


Figure 9: A typical OTSL 5ω camera image for the AA_OTS_PILOT_CL loop (left). The Crosshairs algorithm is used to determine the left and right parallel opposing edges as well as the top and bottom parallel opposing lines or dark bands in the image. The center of the line and edge positions are returned as the beam and reference center locations. The output mask of the adaptive beam location algorithm is shown on the right.

The beam edges and reference lines are found initially using the adaptive beam location algorithm and then further refined with the crosshairs algorithm. The crosshairs algorithm has been used with great success in many loops in the NIF automated alignment control system [8-10]. Crosshairs has proven to be tolerant to object rotations, noise, changes in illumination and target object properties such as line thickness.

All images initially undergo a process to quickly identify unacceptable or 'off-normal' images. The off-normal processor is an image analysis routine that contains a suite of selectable tests each of which analyze the image and classify it as a good or bad image. The tests are limited to basic or common image errors such as an image that is unacceptably dim or blank, missing the beam, all-white, etc.

Adaptive Beam Estimator Algorithm

After successful off-normal processing, an initial beam location estimate mask is found with the adaptive beam location algorithm. This algorithm employs an iterative intensity level process to create an optimize mask for the desired image object(s) [11]. To begin, the image is filtered with a non-linear edge preserving filter[12]. All objects at each intensity level are identified and tested for a set of criteria, for example: size, ranging, signal to noise of objects found, and object location in the image.

The resulting mask created is seen in Fig. 9. The edges of the mask are used to identify the initial, general locations of the beam and reference features. The edges are used to identify sub-image areas to locate initial reference lines or beam edge locations. Sub-image boundaries can be seen outlined in green in the lower right plot in Fig. 10.

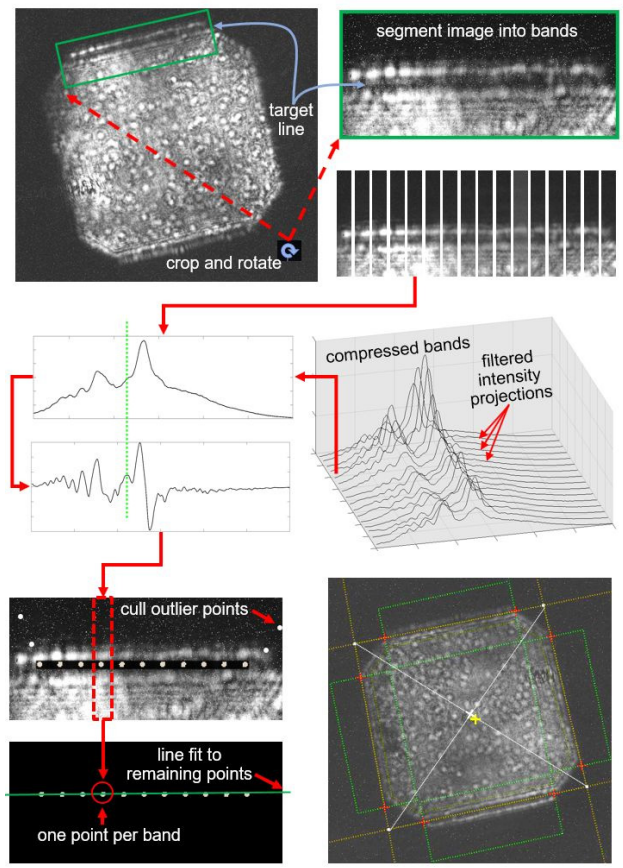


Figure 10: Crosshairs is executed by first rotating the sub-image to orient the target line or edge (top). The image is then segmented into orthogonal slices or 'detection bands' (lower top right). The bands are transformed into a set of intensity vectors by calculating the mean projection for each band (middle). The intensity vectors are then

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL01

processed and filtered individually to identify the edge or line center location for that band (middle left). This results in a set of points, one point per band (bottom left). A series of tests on the resulting set of points is performed to remove outliers. Finally, a least squares linear fit of the remaining points is used to determine the optimal target feature fit (green lines). Final beam location is seen as the yellow + and the reference as the white x and are used to guide beam alignment.

Crosshairs Algorithm

Resulting edge and line locations are then processed and refined with the crosshairs algorithm. A diagram of an example of the complete crosshairs process is illustrated in Fig. 10. Crosshairs is executed by first rotating the sub-image to orient the target line or edge. The image is then segmented into orthogonal slices or 'detection bands. The bands are transformed into a set of intensity vectors by calculating the mean projection for each band. The intensity vectors are then processed and filtered individually to identify the edge or line center location for that band. This results in a set of points, one point per band. A series of tests on the resulting set of points is performed to remove outliers. Finally, a least squares linear fit of the remaining points is used to determine the optimal target feature fit. Uncertainty for any given line estimate is calculated and reported using the mean deviation of the points used in the line fit. The uncertainty is used to abort the alignment if the deviation does not meet the loop requirements. Final beam and reference locations can be seen in Fig. 10 as the yellow + and the reference as the white x and are used to guide beam alignment. Optics adjustments are made until they are co-located and thus alignment for the AA_OTS_PILOT_CL loop is complete.

CONCLUSION

This paper describes the image processing associated with several OTSL alignment loops. In the first two examples, template-based matched-filtering approach was utilized. The last algorithm uses an adaptive beam location algorithm and then is further refined with the crosshairs algorithm.

ACKNOWLEDGEMENTS

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This paper is released as LLNL-CONF-827599-DRAFT. The author wishes to acknowledge the help from Alexandra Saabye for helping with the review and release process.

REFERENCES

[1] A. L. Kritcher, A. B. Zylstra, D. A. Callahan, O. A. Hurricane, *et al*., "Achieving record hot spot energies with large HDC implosions on NIF in HYBRID-E," *Phys. Plasmas* 28, 072706, 2021, doi: 10.1063/5.0047841

[2] S. C. Burkhart, E. Bliss, P. Di Nicola, D. Kalantar, R. Lowe-Webb, *et al*., "National Ignition Facility System Alignment", *Applied Optics*, Vol. 50, Issue 8, pp. 1136-1157 (2011)

[3] J. Galbraith, P. Datte, S. Ross, G. Swadling, S. Manuel, B. Molander, B. Hatch, D. Manha, M. Vitalich, B. Petre, "Design of an Optical Thomson Scattering diagnostic at the National Ignition Facility," *Proc. SPIE 9966, Target Diagnostics Physics and Engineering for Inertial Confinement Fusion V*, 99660E, 2016.

[4] A. Awwal, "Alignment of pointing beam in the Optical Thomson Scattering Laser at the National Ignition Facility," *SPIE Proc. 11841, Optics and Photonics for Information Processing* XV; 118410I (2021), doi.org/10.1117/12.2596187

[5] A. A.S. Awwal, E. Bliss, G. Brunton, V. Miller, R. R. Leach, V. Miller, R. Lowe-Webb, R. Roberts, and K. Wilhelmsen, "Centroid stabilization for laser alignment to corner cubes: designing a matched filter," *Applied Optics*, Vol. 56(1), pp. A41-A51, 2017.

[6] A. Awwal, "Alignment of diffraction features in the Optical Thomson Scattering Laser at the National ignition Facility," *SPIE Proc. 11666, High Power Lasers for Fusion Research VI*; 116660D (2021) doi.org/10.1117/12.2588507

[7] A. A. S. Awwal, M. A. Karim, and H. K. Liu, "Machine Parts Recognition Using a Trinary Associative Memory," *Optical Engineering*, Vol. 28, pp. 537-543, 1989.

[8] R. R. Leach Jr., A. Awwal, E. Bliss, R. Roberts, M. Rushford, K. Wilhelmsen, and T. Zobrist, "Analysis of the confluence of three patterns using the Centering and Pointing System (CAPS) images for the Advanced Radiographic Capability (ARC) at the National Ignition Facility," *Proc. SPIE 9216, Optics and Photonics for Information Processing VIII*, 92161Q (14 October 2014).

[9] R. R. Leach Jr., A. A. S. Awwal, S. J. Cohen, R. R. Lowe-Webb, R. S. Roberts, J. T. Salmon, D. A. Smauley, K. Wilhelmsen, "Alignment mask design and image processing for the Advanced Radiographic Capability (ARC) at the National Ignition Facility," *SPIE Proc. 9598*, 959819 (2015).

[10] R. R. Leach Jr., A. A. S. Awwal, R. Lowe-Webb, V. Miller-Kamm, C. Orth, R. Roberts, K. Wilhelmsen, "Image processing for the Advanced Radiographic Capability (ARC) at the National Ignition Facility," *Proc. SPIE 9970, Optics and Photonics for Information Processing X*, 99700M (14 September 2016).

[11] A. A. S. Awwal, R. R. Leach Jr., "Image processing strategies and multiple paths toward solutions," *Proc. SPIE 10751, Optics and Photonics for Information Processing XII*, 107510R (7 September 2018).

[12] M. Kuwahara, K. Hachimura, S. Eiho, M. Kinoshita, "*Digital Processing of Biomedical Images*," Plenum Press, pp. 187-203, New York, NY, 1976.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

A FRAMEWORK FOR HIGH LEVEL MACHINE AUTOMATION BASED ON BEHAVIOR TREES

G. Gaio†, P. Cinquegrana, S. Krecic, G. Scalamera, G. Strangolino, F. Tripaldi, M. Trovò, L. Zambon, Elettra-Sincrotrone Trieste S.C.p.A., Trieste, Italy

Abstract

In order to carry out complex tasks on particle accelerators, physicists and operators need to know the correct sequence of actions usually performed through a large number of graphical panels. The automation logics often embedded in the GUIs prevents its reuse by other programs, thus limiting the level of automation a control system can achieve. In order to overcome this limit, we have introduced a new automation framework for shifting the logics from GUIs to server side, where simple tasks can be easily organized, inspected and stacked up to build more complex actions. This tool is based on Behavior Trees (BT) which has been recently adopted in the gaming industry for in-game AI player opponents. They are able to create very complex tasks composed by simple decoupled self-contained tasks (nodes), regardless how they are implemented. The automation framework has been deployed in the Tango-based control systems of Elettra and FERMI to implement autonomous operations. A dedicated Qt GUI and a web interface allow to inspect the BTs and dynamically go through a tree, visualize the dependencies, monitor the execution and display any running action.

INTRODUCTION

In Elettra and FERMI, a synchrotron light source and a free electron laser located in Italy, it is usual for control room operators or physicists to manually perform long sequences of operations to configure the accelerators in the desired state. These procedures are prone to errors and heavily dependent on the skills of the operators. Over time, many institutes have developed frameworks to automate these lengthy procedures [1,2,3,4]. Therefore, the framework we are going to present in this article is not an absolute novelty.

To be successful a framework must be easy to use, robust and adopted by as many people as possible. These concepts were the basis for the development of this new framework. The originality lies in inheriting the modularity, flexibility and robustness of the Behavior Trees (BT).

Behavior Trees

BTs are very efficient in modelling what an Artificial Intelligence (AI) algorithm can do. They allow designers to define very low-level tasks and combine them to create the set of high-level tasks that the developer wants available to the AI application.

The Behavior Tree is a directed rooted node tree where the internal nodes are called “control flow nodes” and leaf nodes are called “execution nodes” (see Fig. 1). Briefly, the execution flow starts from a root node and go through the tree down to the leaves. The main control flow node is the *sequence node* that can run in parallel or in series to other *sequence nodes* or *action nodes*. An *action node* executes a task and returns to its parent a *success*, *running* or *failure* state. The *sequence* node returns *success* to its own parent if all its children return *success*. The *sequence node* can be configured as *fallback node* to return *success* when at least one of its children return *success*. A *condition node* returns *success* or *failure* depending on the evaluating condition. A *decorator node* can invert a *failure* state into *success* and vice-versa.

There is no canonical implementation of BTs. They are very flexible and suitable to be customized for any application, whether it is an AI algorithm in a computer game [5] or in an Unmanned Aerial Vehicle [6].

Each node executes an instruction after receiving a tick from its parent. In our implementation this aspect has been neglected. The *Action* node start to executes the task at the first tick and completes procedure detached from any external timing signal.

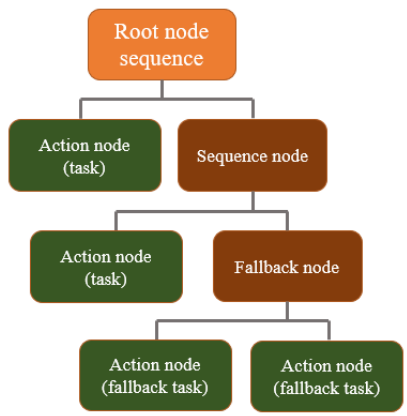


Figure 1: Example of Behavior Tree structure.

THE SEQUENCER

A sequencer (corresponding to a node of the BT) is a Tango device. The core of the sequencer is the sequence which is written in a basic homemade language that has to implement the operations that are performed manually by operators. The language implements macros containing *if/else* conditions and *read/set* instructions, and support basic arithmetic and bitwise operations.

† email: giulio.gaio@elettra.eu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

The sequence is stored in the sequencer Tango device property or in a text file (see Fig.2). The loading of the sequence is carried out during the Tango device initialization. After that, the sequencer device executes the sequence immediately or when a *start* command is received.

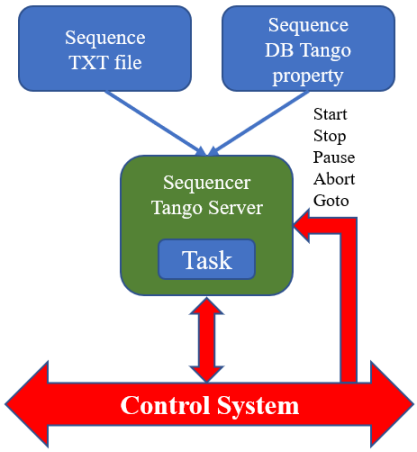


Figure 2: Sequencer device architecture.

The Tango server executes the sequence instructions (commands and read/write attributes) row by row. Each row is identified by a step number (see Fig. 3). For each step a mandatory *goto* statement specifies the next step to execute. The sequence ends after evaluating the last step or can last forever.

The sequencer states are:

- OFF: sequencer is not running (success state)
- RUNNING: sequencer is executing the sequence code
- FAULT: sequencer has terminated due to an exception
- WARNING: sequence parsing error
- STANDBY: the sequence is paused

Internal Variables

The sequence can instantiate internal variables (ex: a loop counter, a threshold, …) which are dynamically mapped into read/write attributes. For simplicity the attribute types are *bool*, *int32*, *int64* and *double,* and can support vectors and images.

Stop Conditions and Exceptions

The sequence stops after receiving an *abort* or *stop* command. While the *abort* command halts the sequence immediately, the *stop* command lets the sequence terminate the running step. In both cases the sequence ends in OFF state, which means *success*.

The sequence can be paused upon receipt of the *pause* command or can execute a single step specified in the *goto* command.

The instruction *goto(-1)* generates an internal exception that stops the sequence in FAULT state. The *goto(N+1)* instruction, where N is the last step number, terminates the sequence in OFF state.

There are four types of exceptions.

- *syntax exception*: it occurs during the sequence loading process in case of a syntax error. The sequencer state turns into ALARM and it cannot be restarted. The status attribute returns the part of the sequence containing the syntax error.
- *global timeout exception*: the sequence execution time takes more than the global timeout; the sequence ends and the device state is set to FAULT.
- *step timeout exception*: the step execution time takes more than its timeout; the sequence stops and the Tango device state changes to FAULT.
- *Tango exception*: when a Tango exception occurs in a command or read/write attribute, the sequence is immediately stopped and the sequence state is set to FAULT. Tango exceptions can be ignored; in this case all *if/else* statements involving Tango read attributes are evaluated as true.

Tango and step timeout exceptions can be caught and the execution flow redirected to a specific step. Similarly, the *Stop* command can be intercepted by the sequencer engine that, instead of ending the sequence, jumps to a predefined "exit" step.

The sequencer can be configured to have its error recognized by the operator. In this case the sequencer requires an *acknowledge* command before receiving a new *start* command.

stepN;[expression];[step description];[error message];[timeout ms.];[catch exceptions]

Example:
step1;read(sr/ps/ch_s1.1/State)==FAULT **?** command(sr/ps/ch_s11/Reset) && **goto(2) : goto(3)**;Reset PS;Error resetting PS;3000
step2;read(sr/ps/ch_s1.1/State) != OFF **?** sleep(1) && **goto(2) : goto(3)**;Waiting OFF state;Timeout waiting OFF state;6000
step3;read(sr/ps/ch_s1.1/State) != ON **?** command(sr/ps/ch_s11/On) && **goto(4) : goto(5)**;Turn PS ON;Error turning ON PS;3000
step4;read(sr/ps/ch_s1.1/State) != ON **?** sleep(1) && **goto(4) : goto(5)**;Waiting ON state;Timeout waiting ON state;6000

Description:
step1; If PS(sr/ps/ch_s1.1) is in FAULT state then Reset and go to step2, otherwise go to step3
step2; if PS(sr/ps/ch_s1.1) is not OFF then sleep one sec and check again (for max 6 sec.), otherwise goto step3
step3; if PS(sr/ps/ch_s1.1) is not ON turn PS ON and go to step4, otherwise go to step5 (exit)
step4;if PS(sr/ps/ch_s1.1) is not ON then sleep one sec and check again (for max 6 sec.), otherwise goto step5 (exit)

Figure 3: Description of the sequencer scripting language. Each code line is composed at least by 5 fields: step index, logical expression, step description, error message and timeout. The catch exception field is optional. The if/else statement is implemented with the ternary conditional operator (? :).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Logging

The sequencer device implements data logging on two levels:

- *high resolution buffer*: a circular buffer of 10000 entries records every instruction executed inside the sequence, any value read or set, and any error or conditional jump with microsecond resolution. It is mostly used by developers during the sequence debugging and, in the near future, analysed by software bots for malfunction prediction.
- *Long-term buffer*: in order to record any modification on machine parameters, each sequencer can store all executed Tango commands and written attributes in a circular buffer of 100000 entries.

The description and error messages encoded in each sequence step are also stored in the long-term buffer.

Currently both buffers are reset after device reinitialization.

Additional Info

In the header of the sequence code it is possible to specify a description of the sequence, the author and the release date.

The sequence device returns the last execution time, the elapsed time from the start, the remaining time till the global timeout and, in case of exception, the last faulty step. Moreover, during sequence execution, the description of the running step is pushed by the sequencer to any event subscriber.

In order to trace dependency between sequence devices, each device returns separately the list of the sequence devices and the list of other Tango devices to which it is connected. This information is useful for monitoring, at a higher (client) level, that the BT respects a Directed Acyclic Graphs (DAG) structure. In fact, no loops are allowed between parent nodes (ex: the root node) and the child nodes.

Templates

In order reuse the same sequence, the concept of template has been introduced. After loading the template from file and before checking its syntax, the Tango sequencer engine replaces keywords with the strings specified in a device property called *wildcards*. The replacing strings are usually Tango device names or constant values.

About fifty templates are available for sequencer developers. Some of them execute specific tasks (ex. recovering a faulty plant), other implement BT flow control, generic algorithms as scans/ramps, basic optimization algorithms (Extremum Seeking, Golden Search) or save/restore procedures. The most used template is the *launcher*. It is the main building block of the BT as it implements both a sequencer and fallback node.

The *launcher* can start up to 64 sub-sequences. Through the use of bit-masks (r/w attributes) it can select groups of sub-sequences:

- to be launched in series and/or parallel.
- to be restarted after a fault for a maximum number of times.
- to be ignored if producing errors and continue in launching the remaining sub-sequences.
- to be stopped at the first fault and abort the launch of the remaining sub-sequences.

Furthermore, a per sequence boolean attribute enables/disable its execution at runtime.

After all sub-sequences have been executed successfully, the *launcher* stops in OFF (or FAULT in case of errors) and its state is acquired by the parent node. When a *launcher* is going to terminate in FAULT state, it is possible to execute, as the last running sequence, a rollback sub-sequence.

Sometimes it is useful to stop, pause or abort the execution of an entire BT. In order to do so, the *launcher* sequencer, if explicitly enabled, can propagate all received commands directly to all child nodes.

Naming and Database Configuration

In the Tango database the naming of the sequences follows the rule:

seq/[*action*]/[*detailed_description*]_[*where*]

As example, the naming of a sequence which performs the power-cycling and communication recovery with a CCD in the Elettra booster is:

seq/reset/ccd_b1.1

Similarly, the sequence calculating the global orbit feedback response matrix in the storage ring is:

seq/calc/gof_rm_sr

A global free property in the Tango database called *sequencer* contains several sub-properties, each one containing a list of root sequence devices grouped by usage. For the Elettra storage ring the sub-property list names are Feedback, InsertionDevices, PowerSupplies, MSCR, Optimization, Recover, BeamOFF.

Another sub-property contained in the *sequencer* global free property called *SuperList* contains the name of all the sub-properties listed above. This information is used by graphical applications to dynamically build up the GUIs for configuration, monitoring and control of the sequence-based operations.

Executing External Programs

The sequencer language supports a restricted group of Tango APIs. For more complex tasks it is possible to execute in the sequence, by means of a c-style “exec” system call, Python, Matlab, Bash and any other type of script.

GRAPHICAL INTERFACE

One of the main problems of automation is making the operator aware of the operations involved in a procedure and understand what is the cause of an error. Both in FERMI and Elettra BTs are made of up to a few hundreds of sub-sequences and the number is constantly growing. In order to provide the operators with flexible command and control tools two graphical panels based on Qt-Cumbia libraries [7] have been developed.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

Sequencer GUI

The panel loads a list of predefined root sequences or, by means of Tango wildcards (ex: seq/*/* selects all sequences in Tango database), composes a dynamic list of all the sequences matching the search criteria. Thanks to the information gathered from the devices and the Tango database, the panel identifies the BT dependencies and builds the BT in a hierarchical list format. The user can inspect the entire BT from the root sequence node down to the leaf nodes (see Fig. 4). A tree structure view is also available. By browsing the hierarchical list, the user gets the description, the state (Tango *state* attribute), the last execution time and the last execution elapsed time of each sequence. In the hierarchical list the states of all non-sequence Tango devices logically connected to the BT are also reported.

The user can start/stop an entire BT or just a part of it. The source code of each sequence can be displayed in a read-only format.

A textual search can be performed simultaneously on the logging buffers of all the sequences belonging to a BT. Furthermore, the panel not only displays live, thanks to the Tango events, the list of errors and settings performed by one BT, but can also collect any action performed by all the sequencers configured in a Tango database.

SeqLauncher GUI

This panel read the *SuperList* free property from the Tango database and creates as many tabs as the number of lists found inside the *SuperList* property. Each tab contains all the root sequences belonging to the corresponding list (see Fig. 5). For each sequence an "OpenGUI" button starts preconfigured legacy applications that ease the monitoring of the BT execution. Similarly, the "CloseGUI" button kills all applications started by the OpenGUI button. This is the panel used by the operators in control room to access directly the sequencers.

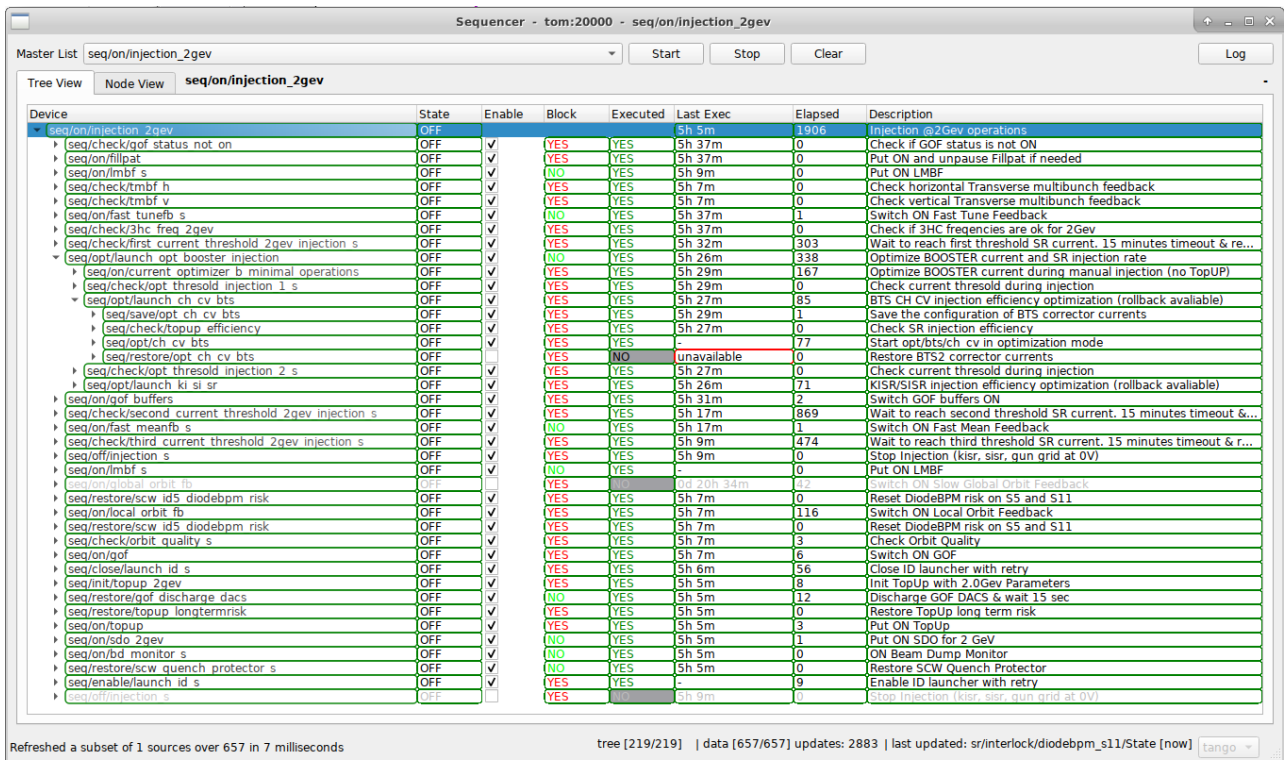
WEB interface

A web interface (see Fig. 6) based on PUMA [8] allows listing all sequence devices installed in FERMI and Elettra. Sequences can be remotely started by users after LDAP authentication.

APPLICATIONS

There are 958 sequencer devices installed in the Elettra Tango database and 995 in the FERMI Tango database.

In Elettra, most of the high-level applications used for the accelerator automation have been replaced by sequencers. There are two main sequencers, the first for recovering the machine from a faulty state, the second for injecting the beam into the storage ring and performing optimizations and orbit correction until a stable beam can be delivered to the beamlines. Thanks to the BT modularity both of them have been embedded in larger sequencer that is able to run the machine autonomously. The work to refine especially the recovery sequencers is ongoing with the goal to face any anomalous situation from which the machine could be automatically recovered.



Device	State	Enable	Block	Executed	Last Exec	Elapsed	Description
seq/on/injection_2gev	OFF				5h 5m	1906	Injection @2Gev operations
seq/check/gof status not on	OFF	✓	YES	YES	5h 37m	0	Check if GOF status is not ON
seq/on/fillpat	OFF	✓	YES	YES	5h 37m	0	Put ON and unpause Fillpat if needed
seq/on/lmbf s	OFF	✓	NO	YES	5h 9m	0	Put ON LMBF
seq/check/tmbf h	OFF	✓	YES	YES	5h 7m	0	Check horizontal Transverse multibunch feedback
seq/check/tmbf v	OFF	✓	YES	YES	5h 7m	0	Check vertical Transverse multibunch feedback
seq/on/fast tunefb s	OFF	✓	NO	YES	5h 37m	1	Switch ON Fast Tune Feedback
seq/check/3hc freq 2gev	OFF	✓	YES	YES	5h 37m	0	Check if 3HC freqencies are ok for 2Gev
seq/check/first current threshold 2gev injection s	OFF	✓	YES	YES	5h 32m	303	Wait to reach first threshold SR current. 15 minutes timeout & re...
seq/opt/launch opt booster injection	OFF	✓	NO	YES	5h 26m	338	Optimize BOOSTER current and SR injection rate
seq/on/current optimizer b minimal operations	OFF	✓	YES	YES	5h 29m	167	Optimize BOOSTER current during manual injection (no TopUP)
seq/check/opt thresold injection 1 s	OFF	✓	YES	YES	5h 29m	0	Check current thresold during injection
seq/opt/launch ch cv bts	OFF	✓	YES	YES	5h 27m	85	BTS CH CV injection efficiency optimization (rollback avaliable)
seq/save/opt ch cv bts	OFF	✓	YES	YES	5h 29m	1	Save the configuration of BTS corrector currents
seq/check/topup efficiency	OFF	✓	YES	YES	5h 27m	0	Check SR injection efficiency
seq/opt/ch cv bts	OFF	✓	YES	YES	-	77	Start opt/bts/ch_cv in optimization mode
seq/restore/opt ch cv bts	OFF		YES	NO	unavailable	0	Restore BTS2 corrector currents
seq/check/opt thresold injection 2 s	OFF	✓	YES	YES	5h 27m	0	Check current thresold during injection
seq/opt/launch ki si sr	OFF	✓	YES	YES	5h 26m	71	KISR/SISR injection efficiency optimization (rollback avaliable)
seq/on/gof buffers	OFF	✓	YES	YES	5h 31m	2	Switch GOF buffers ON
seq/check/second current threshold 2gev injection s	OFF	✓	YES	YES	5h 17m	869	Wait to reach second threshold SR current. 15 minutes timeout &...
seq/on/fast meanfb s	OFF	✓	NO	YES	5h 17m	1	Switch ON Fast Mean Feedback
seq/check/third current threshold 2gev injection s	OFF	✓	YES	YES	5h 9m	474	Wait to reach third threshold SR current. 15 minutes timeout & r...
seq/off/injection s	OFF	✓	YES	YES	5h 9m	0	Stop Injection (kisr, sisr, gun grid at 0V)
seq/on/lmbf s	OFF	✓	NO	YES	-	0	Put ON LMBF
seq/on/global orbit fb	OFF		YES	NO	0d 20h 34m	42	Switch ON Slow Global Orbit Feedback
seq/restore/scw id5 diodebpm risk	OFF	✓	YES	YES	5h 7m	0	Reset DiodeBPM risk on S5 and S11
seq/on/local orbit fb	OFF	✓	YES	YES	5h 7m	116	Switch ON Local Orbit Feedback
seq/restore/scw id5 diodebpm risk	OFF	✓	YES	YES	5h 7m	0	Reset DiodeBPM risk on S5 and S11
seq/check/orbit quality s	OFF	✓	YES	YES	5h 7m	3	Check Orbit Quality
seq/on/gof	OFF	✓	YES	YES	5h 7m	6	Switch ON GOF
seq/close/launch id s	OFF	✓	YES	YES	5h 6m	56	Close ID launcher with retry
seq/init/topup 2gev	OFF	✓	YES	YES	5h 5m	8	Init TopUp with 2.0Gev Parameters
seq/restore/gof discharge dacs	OFF	✓	NO	YES	5h 5m	12	Discharge GOF DACS & wait 15 sec
seq/restore/topup longtermrisk	OFF	✓	YES	YES	5h 5m	0	Restore TopUp long term risk
seq/on/topup	OFF	✓	YES	YES	5h 5m	3	Put ON TopUp
seq/on/sdo 2gev	OFF	✓	NO	YES	5h 5m	1	Put ON SDO for 2 GeV
seq/on/bd monitor s	OFF	✓	NO	YES	5h 5m	0	ON Beam Dump Monitor
seq/restore/scw quench protector s	OFF	✓	NO	YES	5h 5m	0	Restore SCW Quench Protector
seq/enable/launch id s	OFF	✓	YES	YES	-	9	Enable ID launcher with retry
seq/off/injection s	OFF		YES	NO	5h 9m	0	Stop Injection (kisr, sisr, gun grid at 0V)

Figure 4: Overview of the BT used for the Elettra storage ring injection at 2GeV. The Root node launches 31 sub-sequences over a total of 219. The total procedure lasted 1906 seconds. Excluding the "waiting tasks", the longest ones during the injection are: automatic optimizations (338 sec.), orbit correction (116 sec.), undulator gaps closing (56 sec.).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

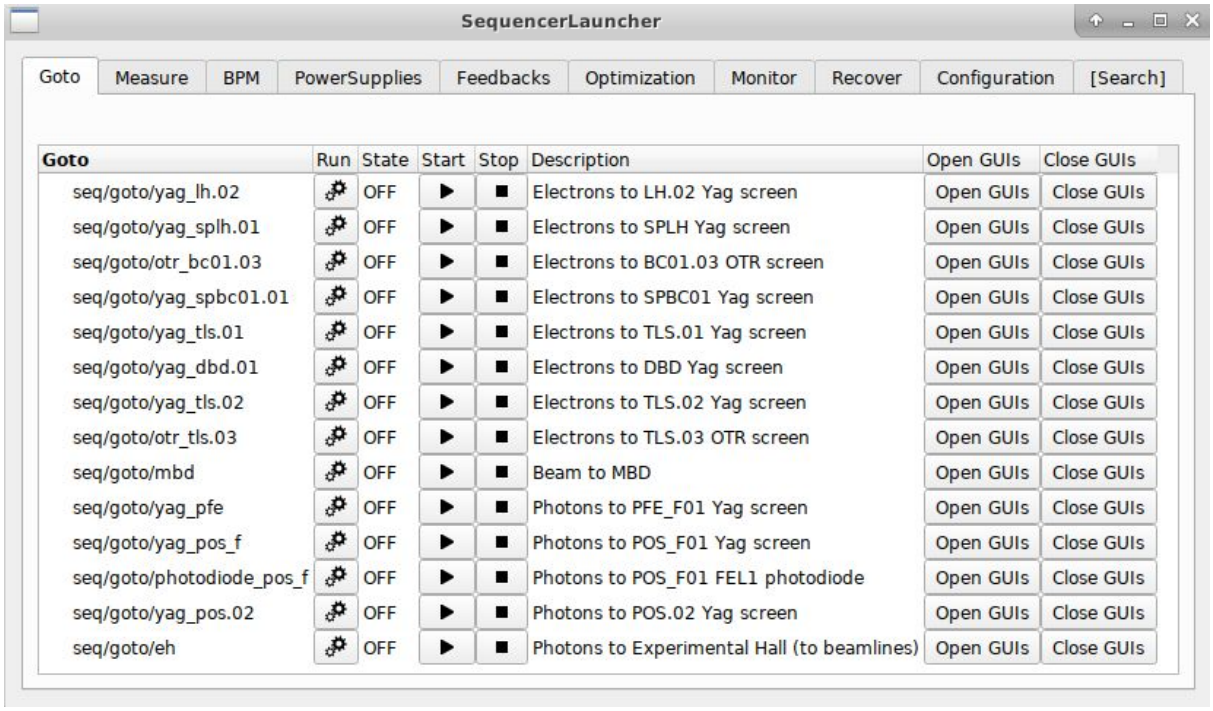


Figure 5: Control panel for driving electron and photon beams to diagnostic stations and beamlines in FERMI.

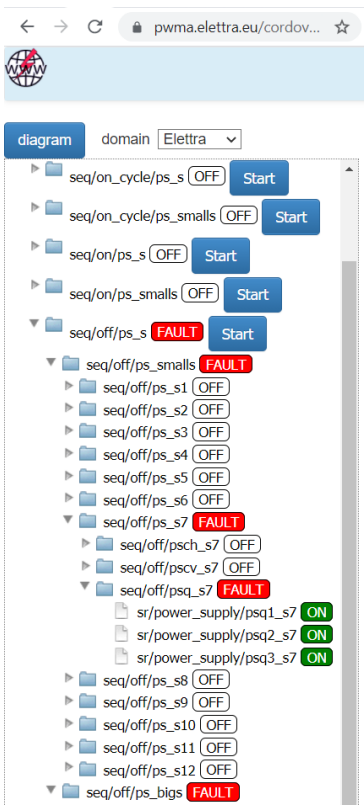


Figure 6: Web based sequence browser.

In FERMI the sequencers have been introduced quite recently. They supervise the online the FEL automatic optimizers and reconfigure the machine for delivering the electron and photon beams to diagnostic stations and beamlines.

CONCLUSION

With the adoption of sequencers, developing automation applications with very complex logics has become much easier than in the past. More people, even non-professional programmers, are actually working on automation, bringing machine autonomous operations to unprecedent levels of complexity.

Thanks to the sequencers, moving logics from GUIs to server applications is possible and convenient: the server-side logics can be recalled by any client in the control system without caring of its implementation. Moreover, encoding in a sequencer a procedure that is normally written in an operator manual or, even worse, known by just one expert, is a clear advantage. Once the optimal sequence is defined and encoded, everyone can perform that task at expert level.

Another advantage in the present Machine Learning era is that, by using sequencers, the machine operations become more deterministic. Machine settings are not polluted by the randomicity of the human actions. As a consequence, any Machine Learning algorithm designed to learn from logs any insight of malfunctions or weird behavior of the machine, will benefit from processing cleaner data.

To start playing with the sequences there is no need of external software except the sequencer Tango server and the Tango control system. For building up more complex applications, the use of the *Launcher* template and the Qt GUIs are strongly suggested.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL02

REFERENCES

[1] D.E. Johnson, R.P. Johnson, "The Colliding Beam Sequencer", in *Proc. 13th IEEE Particle Accelerator Conference (PAC89),* Chicago, IL USA, March 1989, doi:10.1364/OE.20.11396

[2] D. Bulfone, F. Potepan, C. Scafuri, "Automizing ELETTRA Operation with One Button Machine", in *Proc. 17th IEEE Particle Accelerator Conference (PAC97),* Vancouver, Canada, May 1997, doi: 10.1109/PAC.1997.751242

[3] J. van Zeijts, T. D'Ottavio *et al.*, "The RIHC Sequencer", in *Proc. of the 2001 IEEE Particle Accelerator Conference (PAC01)*, Chicago, IL USA, June 2001, paper MPPH008, pp. 782-784, doi: 10.1109/PAC.2001.986473

[4] V. Baggiolini, R. Alemany-Fernandez *et al.*, "A Sequencer for the LHC ERA", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09),* Kobe, Japan, Oct. 2011, paper THC003, pp. 670-672.

[5] Y. A. Sekhavat, "Behavior Trees for Computer Games", *International Journal of Artificial Intelligence Tools*, 26(2), Jan. 2017, doi: 10.1142/S0218213017300010

[6] P. Ögren, "Increasing Modularity of UAV Control Systems using Computer Game Behavior Trees", in *Proc. AIAA Guidance, Navigation, and Control Conference (MGNC12),* Minneapolis, Minnesota USA, Aug. 2012, doi: 10.2514/6.2012-4458

[7] G. Strangolino, "Cumbia: A New Library for Multi-Threaded Application Design and Implementation", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17),* Barcelona, Spain, Oct. 2017, paper TUPHA174, pp. 830-834, doi: 10.18429/JACoW-ICALEPCS2017-TUPHA174

[8] G. Strangolino, L. Zambon, "Canone3: a New Service and Development Framework for the Web and Platform Independent Applications", *presented at the 18th Int. Particle Accelerator Conf. (ICALEPCSC'21)*, FRAR02, Oct 2021, this conference

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL03

THE STATUS OF FAST OBIT FEEDBACK SYSTEM OF HEPS

P. Zhu†[1], D. P. Jin[1], Y. L. Zhang[1], Z. X. Xie[1], Z. Lei[1], D. Y. Wang[1],
Y. C. He[1], K. J. Xue[1], W. Xuan[1], L. Wang[1]
Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China
[1]also at Spallation Neutron Source Science Center, Dongguan, China

Abstract

The new fast orbit feedback (FOFB) system, a typical multiple-input and multiple-output (MIMO) system, is essential for the storage ring of High Energy Photon Source (HEPS). In order to reduce overall latency and achieve more than 500Hz bandwidth, the FOFB system adopt 16 sub-stations with the same hardware and software function to obtain bias-data from the beam position monitors (BPMs) data using 2.38Gbps in the SFPs and send correct-data to the fast corrector power supplies using a serial point-to-point link around the storage ring, and each sub-station share BPMs data with daisy-chained using of 10Gbps in the SFPs. It is optimized to calculate the large matrix based on the singular value decomposition (SVD) taking the digital signal processing (DSP) modules of V7 field programmable gate array (FPGA) with parallel pipeline. For performance tuning and additional flexibility when adding or removing BPMs and fast corrector power supplies, or downloading new matrix, we implement an independent ethernet controller for remote operation. This article presents details on the design and implementation of the FOFB system, and also the future improvements.

INTRODUCTION

High Energy Photon Source (HEPS), a high-performance and high-energy synchrotron radiation source, is one of the major national scientific and technological infrastructures, mainly composing of accelerators, beam lines and experimental stations, supporting facilities, etc, such as show in Figure 1. HEPS will be an original and breakthrough in the fields of basic science and innovative research.

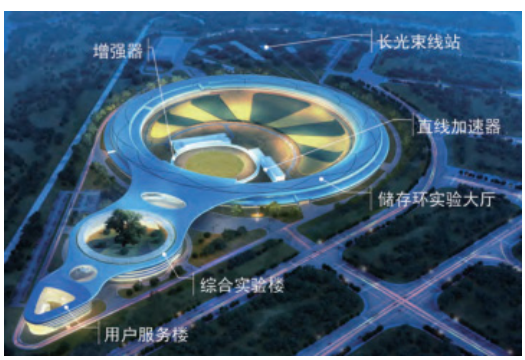


Figure 1: Layout of HEPS.

The size of modern light sources is getting smaller and smaller, and the brightness is getting higher and higher. The performance of the synchrotron radiation photons that finally reach the sample is closely related to the orbital stability of the electron beam. According to the practice of major international laboratories, the stability of the position should be about 10% of the bunch size. For HEPS, the brightest fourth-generation synchrotron radiation source in the world, the bunch vertical emittance should be on the level of 10pm.rad, which corresponds to the vertical bunch size about less than 3μm. Therefore, at the short straight section, the stability of the beam position should be better than 0.3μm.

REQUIREMENT

There are many factors that affect the stability of the beam orbit, including the stability of the magnet power supply, ground vibration, temperature effects, etc. According to ring accelerator physical design and requirements, there are many beam position monitors (BPMs) to monitor the orbit and many correctors to correct the orbit in the storage ring. In order to suppress interference and keep the beam orbit stable, it is we adopt a high-intensity and high-speed orbit feedback system, a typical multiple-input and multiple-output (MIMO)system, to achieve long-term stable operation of the light source based on singular value decomposition (SVD) commonly [1]. The correction algorithm is based on an SVD of the orbit response matrix:

$$\Delta\overline{X} = R\,\Delta\overline{\theta} \quad and \quad R = USV^T$$

$$\Delta\overline{\theta} = VS^{(-1)}U^T\,\Delta\overline{X}. \tag{1}$$

Where $\Delta\overline{X}$ is error that the current orbit is compared to the golden orbit, U and V are matrices whose columns form an orthogonal basis in BPM(X) and corrector magnet θ space, S is the diagonal matrix of singular values. The proportional-integrator (PI) control algorithm operates in this diagonal space and the relevant parameters can be adjusted for each mode separately [2]. All multiplication and calculations stages are done for all eigenmodes at one cycle. Therefore, it is necessary to comprehensively consider the whole time consuming, including the speed of BPMs data acquisition, the delay of global BPMs data distribution and calculation, and other factors. According to accelerator physical design and requirements, the fast orbit feedback system should optimize strategies to achieve the low latency response with BMPs and fast corrector power supplies, which is essential to achieve a certain effective bandwidth range for the stable orbit.

† zhup@ihep.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL03

ARCHITECTURE

The main part of the HEPS is an ultra-low emission electron storage ring, composing of 48 identical hybrid 7BAs cell. According to the accelerator design, there are 12 BPMs and 8 fast corrector power supplies in each cell.

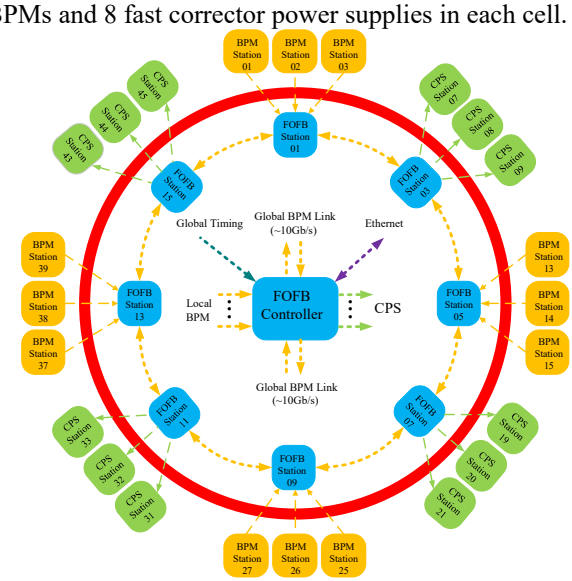


Figure 2: Layout of FOFB architecture.

The storage ring of HEPS, a kilometre scale ring, in order to reduce overall latency and consider feasibility study, we adopt the architecture of daisy-chain based on 16 sub-stations, each sub-station is responsible to interact with local BPMs and fast corrector power supplies. Figure 2 and Figure 3 are shown the layout of FOFB architecture and connection between sub-stations.

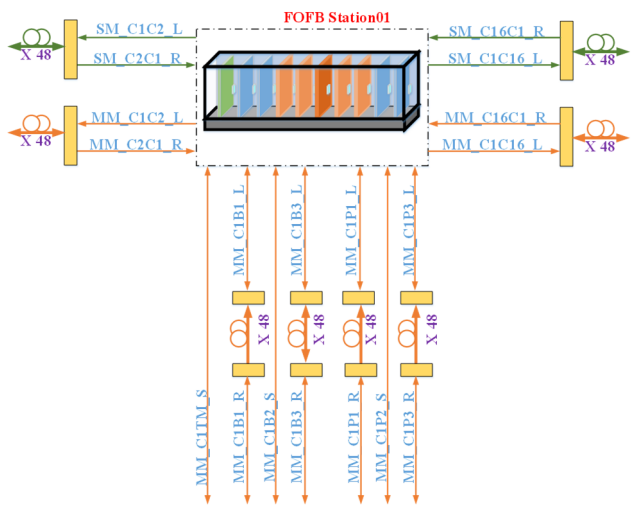


Figure 3: Layout of connection between sub-stations.

Detailly, each sub-station is design and implement to obtain bias-data from BPMs using 2.38Gbps in the SFPs and send correct-data to the fast corrector power supplies using a serial point-to-point link, and each sub-station share BPMs bias-data using of 10Gbps in the SFPs. In order to ensure synchronization with the global beam, all the above working sequence is strictly in accordance with the global clock. For performance tuning and additional flexibility when adding or removing BPMs and fast corrector power supplies, or downloading new matrix, an embedded Ethernet controller is designed as a full hardwired TCP / IP providing internet connectivity to sub-station by using Serial Peripheral Interface (SPI). Figure 4 is shown the hardware architecture of FOFB sub-station.

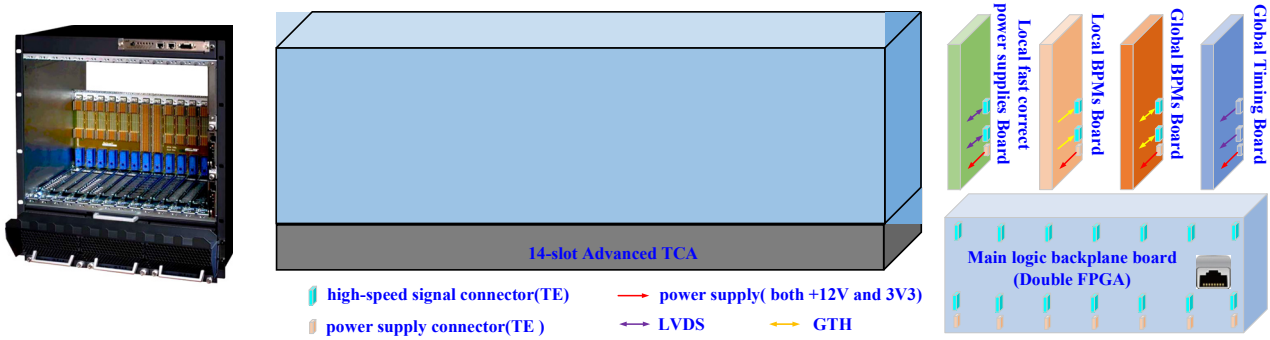


Figure 4: The hardware architecture of FOFB sub-station.

As show in Figure 4, in pursuit of the stable performance with high-speed data transmission, the hardware system of sub-station is only based on ATCA mechanical architecture, but custom the main logic backplane board and the front boards. Furthermore, the sub-station hardware mainly includes:

- 1 main logic backplane board;
- 4 front boards for local BPMs, one of which is reserved for XBPM for beam line;
- 1 front boards for global BPMs;
- 4 front boards for local fast corrector power supplies;
- 1 front boards for global Timing;

The main logic backplane board and the front boards use high-speed TE signal connectors to interact mutually, and providing the front board power supply.

HARDWARE DESIGN

Main Logic Backplane Board

The self-developed main logic backplane board followed the ATCA mechanical size is one of the core board of FOFB. Considering the stability and reliability for long-term operation and future upgrade, the backplane of FOFB is adopt a large-capacity, high-density architecture, as

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

shown Figure 5. We have completed the whole schematic and PCB design, including:

- FPGA chip: Xilinx/XC7VX690T;
- High-speed: 80 channels of GTH transceiver @10Gb/s (3 front boards@16 channels for local BPM data collect, 1 front boards@16 channels for global BPM data transmission);
- Clock: embedded timing / on-board 100MHz crystal clock / LEMO clock output;
- LVDS: 14 slots @20 channels;
- Ethernet: W5500iso;
- Power: +12V power supply.

… …

A total of 22 layers (thickness 2.8mm) of the main logic backplane board PCB are preliminarily drawn up, as shown in Figure 7, which is divided into signal layer and plane layer:

(1) Signal layer:
Top/L3/L5/L7/L9/L14/L16/L18/L20/Bottom, the thickness of copper foil is 0.5oz;

(2) Plane layer:
L2/L4/L6/L8/L10/L11/L12/L13/15/L17/L19/L21, the thickness of L10, L11, L12, L13 copper foil is 2oz used for 1V@100A power supply rail and GND, other lays are all 1oz.

Note: FPGA and other modules are mainly placed on the top layer, which faces the rear of the chassis.

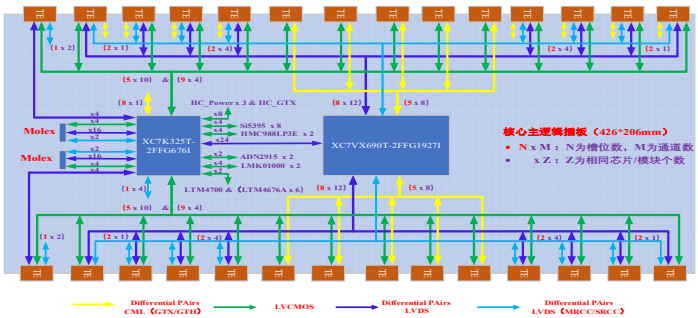


Figure 5: The diagram of signal flow.

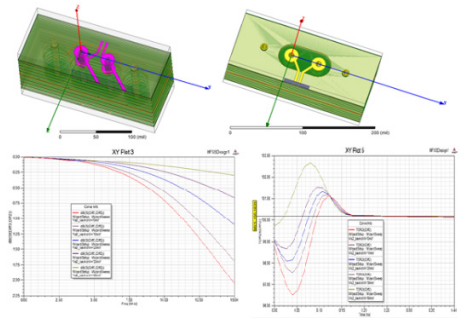


Figure 6: The diagram of via simulation.

Via acts as conductors (PCB) in connecting traces on different layers. In the case of low frequencies, via will not affect signal transmission. However, as the frequency increases (above 2.5 GHz) and the rising edge of the signal becomes steep, via cannot be only simply regarded as a function of electrical connection, but must be carefully considered for signal integrity, which appear as breakpoints with discontinuous impedance on the transmission line, causing signal reflections, as shown in Figure 6.

The power distribution network (PDN) characterizes the change of load size and load frequency, as shown in Figure 8. Its parasitic parameters must be small enough to ensure that in a certain frequency range to ensure that the power supply voltage meets the working voltage range of the load under the maximum current scenario. However, the voltage of the load transmitted by the PDN is no longer equal to the output of the power supply. Once the voltage obtained by the load is lower than the working demand of the load, it will cause unpredictable failure of the load. When the steady-state current of the load reaches a certain value, the supply voltage of the power supply meets the working voltage range of the load, which is the low frequency part of the PDN.

The FOFB main logic backplane is designed with multiple independent chips based on the IIC or SPI bus. In order to monitor the operating status of the chips in real-time, it is necessary to read the operating parameters and measured information through the interface of a FPGA chip, released by ethernet timely. IIC or SPI bus mainly includes 5 parts: the fast corrector power supply front board, the local/global BPMs data front board, the core main logic backplane and the ATCA slot, all of which have been design detailed.

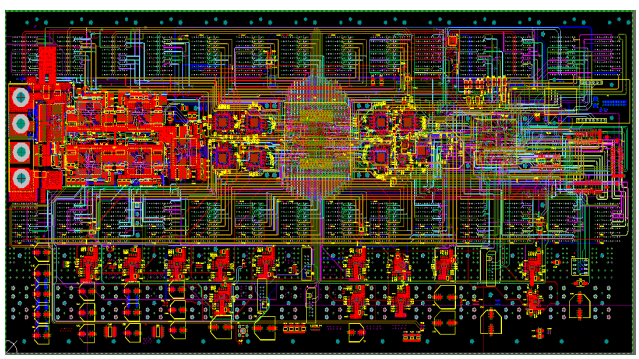


Figure 7: The diagram of PCB.

Target Impedance	Units	Value
Supply Voltage (Min)	V	1
I max	A	100
Transient Current	%	50
Vripple (+/-)	%	5
Frequency target	MHz	250
Ztarget = ΔV / ΔI	Ω	0.0010

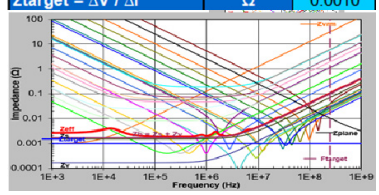


Figure 8: The diagram of capacitance simulation.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAL03

Front Board

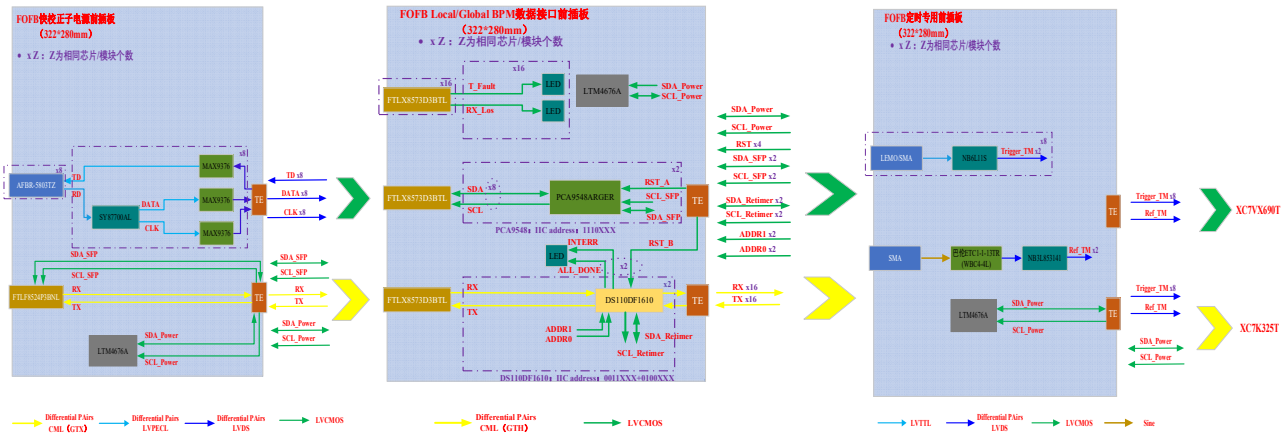
Figure 9 is shown the diagram of signal and function of the front board.

The fast corrector power supply front board is mainly responsible for outputting current settings for power supply substations, and the data transmission rate is about 50Mb/s.

The local BPMs data front board is mainly responsible for collecting 3*12 groups of 32bit effective data, the rate is about 2.38Gb/s. The global BPMs data front board is mainly responsible for transmitting the BPMs data collected by this substation to the global BPMs data transmission link, and where is obtained other FOFB substations, and the rate is about 9.5Gb/s.

The local BPMs data front board and the global BPMs data front board adopt the same hardware based on the ATCA standard (322mm*280mm).

The dedicated timing front board is mainly responsible to receive the timing trigger signal and reference clock signal from global timing and sent to the main logic backplane board. The FOFB system is controlled by the 22kHz timing signal, reset signal, system synchronization start signal, system synchronization stop signal, and loop injection signal sent by the timing system.



(a) (b) (c)

Figure 9: The diagram of signal and function of the front boards.

LOGIC DEVELOPMENT

Data Calculation Module

The data arithmetic module adopts a pipeline design, which is advantage that it can accurately control the data to reach a certain position at a certain time and reduce the delay at the same time, as shown in Figure 10.

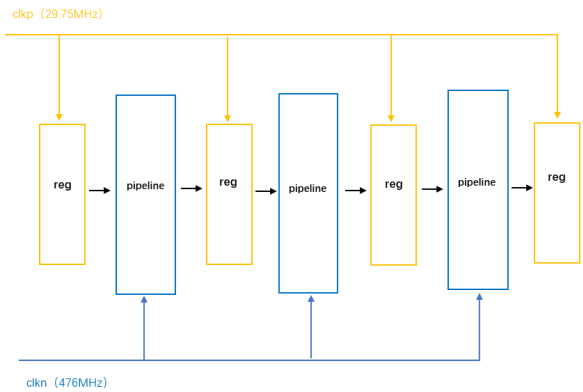


Figure 10: The diagram of pipeline design.

Data Storage Module

Because the amount of parameter data that the system needs to store is very large, so it is necessary to call the Block Memory resource in the FPGA for data storage. Considering the design goal of FOFB low latency, BRAM is used in the memory module design to ensure that the data is written and read in one clock cycle to reduce latency.

CONCLUSION

In order to reduce the system response time as much as possible and improve the effective feedback bandwidth of the FOFB system, we propose a new hardware architecture that can quickly obtain all BPMs data in the storage ring, and quickly complete large data calculations, and send the setting to the fast corrector power supply in real-time. It is shown that the architecture can achieve an effective bandwidth of more than 500 Hz, which also meets the main requirements of the HEPS project.

ACKNOWLEDGEMENTS

Thank you for all the leaders, teachers and colleagues of the research group of HEPS, especial members of FOFB.

REFERENCES

[1] S. Kongtawong, *et al.*, "Recent improvements in beam orbit feedback at NSLS-II," NIM. Methods Phys. Res. A 976 (2020) 164250.

[2] Y. Tian, *et al.*, "NSLS-II Fast Orbit Feedback System", in *Proc. ICALEPCS2015*, Melbourne, Australia, Oct. 2015, pp. 34-37. doi:10.18429/JACoW-ICALEPCS2015-MOC3005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

THE TANGO CONTROLS COLLABORATION STATUS IN 2021

A. Götz, R. Bourtembourg, D. Lacoste, N.Leclercq, ESRF, Grenoble, France
S.Rubio, C. Pascual-Izarra, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
V. Hardion, B.Bertrand, MAXIV Sweden, Lund, Sweden
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
P.P. Goryl, M. Liszcz, S2Innovation, Kraków, Poland
A.F. Joubert, J.Venter, SARAO, Cape Town, South Africa
G. Abeille, SOLEIL, Gif-sur-Yvette, France
T. Braun, Byte Physics, Berlin, Germany
G. Brandl, Forschungszentrum Jülich, Garching, Germany

Abstract

The Tango Controls collaboration has continued to grow since ICALEPCS 2019. Multiple new releases were made of the stable release V9. The new versions include support for new compiler versions, new features and bug fixes. The collaboration has adopted a sustainable approach to kernel development to cope with changes in the community. New projects have adopted Tango Controls while others have completed commissioning of challenging new facilities. This paper will present the status of the Tango-Controls collaboration since 2019 and how it is helping new and old sites to maintain a modern control system.

INTRODUCTION

Tango Controls is a software toolkit for building object oriented control systems. It has been adopted at a large number of sites around the world either as the main toolkit for their control system or for a sub-system or commercially acquired systems. A growing number of commercial products and control systems are now based on Tango Controls. A few commercial companies offer paying support for anyone needing help in integrating Tango Controls into their system.

The main objectives of kernel developments for Tango Controls since 2017 has been to consolidate the continuous integration for all major platforms, maintain the Long Term Support version V9, i.e. bug fixing and add features which are strongly needed by the community but do not break compatibility with V9 release, improve the web development platform, continue to improve the documentation and website, and prepare the next major release of Tango (V10). This paper summarises these developments.

COLLABORATION

The Tango Controls Collaboration is in charge of ensuring the sustainability of Tango Controls. It currently has 11 members who contribute financially and in-kind to the maintenance of Tango Controls as a modern reliable controls toolkit for small and large facilities. In addition to the collaboration members a number of individuals and some companies contribute new developments to the ecosystem e.g. see section Rust binding below.

The Tango Controls Collaboration contract has been in operation since 5 years. The initial collaboration contract ended in 2020. All partners agreed the collaboration was fulfilling an essential role for the sustainability of Tango and should be continued for another 5 years. All partners signed the new contract running from 2021 to 2025. The new contract maintains the objectives and missions of the previous one i.e. all members provide the same financial contribution to maintaining the kernel, while some partners contribute in-kind resources too. The ESRF is in charge of sub-contacting tasks on behalf of the other members. A major change in sub-contracting took place in 2021 with a Call For Tender for sub-contractors who could provide services to Tango for the next 3 to 5 years. After a selection process 4 companies were chosen based on their competence and knowledge of the Tango kernel. Setting up contracts for 3-5 years will help ensure the sustainability of Tango controls. The new collaboration contract foresees a rotation of the role of coordinator amongst the members every year.

New Projects

New projects continue to adopt Tango as their controls toolkit. Some examples of major new projects are the LOFAR 2.0 project (see [1]), the JINR 200 MeV LINAC (see [2]), the PEPC plasma electrode Pockels cell (see [3]) to mention a few. Some of the large projects which are based on Tango have completed successfully, for example the ESRF-EBS, the first 4th generation storage ring (see [4] and [5]). Other large projects like the SKA are ramping up to full speed and are already well advanced in their developments and will soon start construction (see [6]). A number of other 4th generation storage rings which will be based on Tango (e.g. ELETTRA, SOLEIL and CELLS) are in the planning phase. The above projects are only a small subset of projects using Tango: due to the way open source code can be downloaded by everyone without registering not all projects are declared or known to the community. They illustrate how vibrant the Tango Controls community is and the strong need to sustain and continue developing the Tango Controls toolkit for the coming decades.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

KERNEL DEVELOPMENT

Migrating from GitHub to GitLab

The tango-controls GitHub organization [7] had been used since 2016 for hosting the various tango source code repositories as well as for managing issues and code review. The Continuous Integration was based on Travis-CI [8], also integrated in GitHub. A change in the terms of use of Travis-CI in December 2020 prompted the migration of the whole tango-controls organization to the GitLab platform. GitLab was found better suited to the open source nature of the Tango collaboration, offered more support for Continuous Integration and the fact that its Open Source version can be installed on-premises (as it is the case in many of the facilities involved in the Tango Collaboration). GitLab therefore offers better protection against potential vendor lock-in. The GitLab.com organisation graciously accepted to host the tango-controls project with gold status free of charge. Gold status allows up to 100 seats for developers of which currently 62 are in use.

The migration is being done on a per-project basis (at the moment of writing, 49 have been migrated, out of 67 originally in the tango-controls organization), and the main code review, continuous integration and issue handling is already done in GitLab.com. In terms of implementation, the migration has been relatively simple thanks to the automated import of GitHub projects provided by GitLab. The only aspects that needed some manual intervention were adapting the CI configuration from Travis-CI to GitLab-CI and the coordination to ensure that the contributors had linked their GitLab and GitHub accounts in order to preserve the cross-references and contribution statistics.

See paper [9] for more details.

C++ Core Library

Part of the TANGO C++ core library development is being subcontracted to Byte Physics and S2Innovation for the TANGO collaboration. The Tango C++ Kernel library cppTango [10] is actively maintained with the excellent expert help of these companies.

Since the last ICALEPCS conference, cppTango 9.3.4 [11] has been released and a new Tango Source Distribution Release [12] has been prepared. This new cppTango release provides several bug fixes for race conditions, memory leaks, issues related to events, compilation warnings and better support of recent cppzmq versions [13]. It is fully ABI and API compatible with the previous 9.3.x cppTango releases. This version still does not require C++11 support from the compiler, allowing it to be built on older platforms, including Debian 7.

Since the introduction of the TANGO 9 Long Term Support, the Tango C++ Kernel developers are working on improving the code quality by refactoring the source code to make it easier to maintain. This work is being done in the cppTango repository main branch from which cppTango 9.4 will be released at some point. cppTango 9.4 will not be binary compatible with cppTango 9.3.x but will still be API compatible with cppTango 9.3.x versions. The main development branch requires at least C++14 and will be compatible with C++17 and C++20 . This branch is being automatically compiled and tested with GitLab CI on Ubuntu 20.04 LTS, Debian 9, 10, and 11 platforms as well as using the latest releases of GCC and Clang. The migration to GitLab was an opportunity to replace SonarCloud services with CI jobs using Clang Static Analyzer [14] and clang-tidy [15] to detect code quality issues. Also a new CI job that generates a code coverage report was added. Additionally, some steps were taken to improve the memory safety and the overall stability of the C++ core library. As a result a set of sanitizer CI jobs, including address [16], undefined behavior [17] and thread sanitizers [18], was enabled for the main branch.

Python Core Library

Python is still enjoying good popularity in the Tango Controls community. This is largely due to the excellent PyTango binding for Tango to C++. It provides a high level python friendly API which makes developing servers and clients extremely easy.

PyTango 9.3.3 has been released in December 2020 [19]. This marks only the second release since the previous review at ICALEPCS 2019 [20], with the 9.3.4 release is expected this year. Fourteen contributors have been working on the project producing more than 150 commits, in total. PyTango 9.3.3 supports the releases 9.3.x of the Tango C++ library - the minor versions are kept in sync. No major updates were required in this period, so it was largely maintenance.

PyTango fully supports both Python 2 and Python 3 language versions, although Python 2 is nowadays officially obsolete and its usage have been already discouraged for new developments. Having compatibility with both language versions has helped during the transition to Python 3 of legacy python-based projects. The biggest frameworks using PyTango, like Taurus and Sardana, are already running in production using Python 3. Older Python tools and device servers (fandango, pytangoarchiving, panic) are currently in the process of migration using GitLab CI/CD for validating the migrated code. Unfortunately, the slow pace at which institutes are able to transition all their code to python 3 means that PyTango needs to maintain Python 2 supports for some years still.

The long running port [20] of the C++ extension code from Boost [21] to pybind11 [22] has not progressed significantly. There were few resources to allocate to this work. As there is no major risk of Boost's Python support becoming obsolete, the Tango Steering Committee has decided to pause this effort.

Future work includes making PyTango available as a binary wheel for Linux. This will significantly simplify installation for users.

Java Core Library

Java core library (aka JTango), which is 100% Java, relies on Jacorb [23] and JeroMQ [24], is now built upon Java 8

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

and Java 11. The latest developments have focused on its quality; by fixing bugs, replacing deprecated Java code with Java 8 code, and adding automatic unit and integration tests. JTango artifacts were deployed automatically to Bintray [25] and manually on Maven Central [26] repositories. Since Bintray has announced its shutdown [27] and we were in the process of migrating to GitLab, the CI/CD pipeline were completely refurbished, to optimise its execution time and to automatically deploy artifacts to Maven Central repository (see details in paper [9] at this conference).

POGO

Pogo is the Tango code generator for device servers. It allows users to define a Tango class model through its graphical user interface and save it in an .xmi file. Starting from this Tango class model, Pogo is able to generate a device server skeleton in C++, Java or Python and the relevant HTML documentation. The code generation part is based on EMF (Eclipse Model Framework) associated with Xtext and Xtend classes [28]. Pogo source code is available on GitLab [29], and binary distribution is maintained on maven central [30].

In the past years Pogo has continued to be improved. Python support greatly evolved, two flavors of python device servers can be generated. With the introduction of PythonHL, device servers code is now more concise, more readable, easier to maintain, and faster to develop.

Latest developments focus on improving Pogo for automation. Current development workflows rely more and more on CI/CD and Pogo needs to integrate and keep up to date with the latest strategies. This has lead to:

- A ready to use docker image is now available. It will speeds up deployment strategy and can be used for automatic testing.
- Improved cmake integration. Modern cmake is the de facto standard for building C++ projects, it simplifies building and managing dependencies.

PACKAGING

RPM

Tango RPM packages used to be built internally by MAX IV and provided to the community via the MAX IV's repository [31]. The Tango SPEC file repository [32] has been moved to the tango-controls group on GitLab. Building is now performed using Copr [33], a build system and infrastructure provided by Fedora. Packages are produced for CentOS 7, CentOS 8, Fedora 32 and 33. The latest version of Tango can be installed on those distributions directly from the Copr repository. RPMs are still available from the MAX IV's repository as well.

Conda

Conda [34] is an open-source solution for dependency and environment management for any language. It is cross-platform and runs on Windows, macOS and Linux. Some packages (pytango, itango, tango-test) were already available using the tango-controls conda channel [35]. Some effort were made during this year to publish those packages and add new ones to conda-forge [36]. conda-forge is a community-led collection of recipes, build infrastructure and distributions for the conda package manager. It is very active and has become the de facto channel when using conda. Using conda-forge has several advantages:

- existing conda-forge infrastructure makes it easy to build on all supported platforms
- it ensures compatibility with conda-forge packages by using the same build environment and baseline software versions defined in conda-forge-pinning [37]
- makes it easier for people to discover tango packages

The current Tango packages available on conda-forge are: tango-idl, cpptango, tango-database, tango-admin, tango-test, pytango and itango. This move was also beneficial to the wider Tango community allowing software like Sardana, which depends on pytango, to be available on conda-forge. Only Linux is currently supported but work for publishing Windows packages is planned.

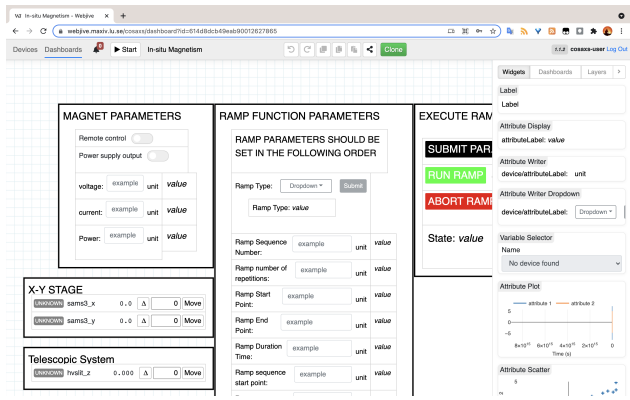


Figure 1: Example of a web interface developed with Taranta - display of status of MAXIV magnets.

WEB TOOLKIT

During the last 2 years the pandemic situation has demonstrated the importance of using web technology in the Tango Control System. There are currently mainstream 2 solutions maintained by the community members, Taranta [38] and Waltz. Taranta (shown in Fig. 1) is now an official tool of the Tango ecosystem and has moved to the tango-control GitLab repository. Following the example of the Tango REST API which has a formal specification (see [39]), a similar project is planned for the specification of the GraphQL Tango gateway in order to allow different back-end implementations (see [40] for example) to connect to multiple front-end applications using GraphQL.

New initiatives have chosen the ubiquitous web browser for their User Interface (UI) like the Solaris Synchrotron [41] who developed the interlock and machine status UI with the Vue.js [42] framework on top of TangoGQL [43].

In the meantime S2Innovation has developed IC@MS a new web UI for the standard Tango Alarm system (see [44]).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

```
use tango_client::*;

let mut dev = DeviceProxy::new("tango://localhost:10000/sys/tg_test/1")?;
let instr = CommandData::from_str("This is a minimal Tango test client.");
let argout = dev.command_inout("DevString", instr)?;
println!("Command exec result: {}", argout.into_string()?);
```

Figure 2: Tango Rust binding example code.

This is an example how the traditional Tango tools are getting a lifting with modern technology.

BINDINGS

Rust Binding

Rust [45] is a fairly new systems programming language. Because it puts a focus on efficiency while ensuring memory and thread safety at compile time, it has the potential to be used, also in the scientific sector, where code is currently being written both in C/C++ and Python, trying to provide the speed of the former together with the usability of the latter. [46] The Tango Rust binding [47] is a Rust library ("crate") that provides Rust code with client access to Tango devices, wrapping the C binding. So far, we have implemented a `DeviceProxy` API up to Tango 8 and rudimentary a `Database` interface. Figure 2 shows a minimal code example calling a method on a TangoTest server. The library is available on Rust's central package server crates.io [48] and requires an installation of the C++ libtango to be present on the system to build against, as well as a C compiler to compile the bundled C binding. It is currently only tested on Linux systems.

TUI

A small text based command line utility have been developed to explore running Tango devices. Devices are presented in a tree structure, from which they can be traversed and selected (see screenshot in Fig. 3). Once a device is selected, their attributes and commands can be seen.

The tool is written in Rust making use of the Rust bindings. The Rust bindings have some limitations that may cause issues when using the tool e.g. enum types are currently not supported. For more details on the tool including its limitations, installation instructions etc. see the project GitHub page [49].

COMMUNITY WORKSHOPS

The 35th Tango Community Meeting was held on September 14 and 15 2021. Despite the pandemic and the remote nature of this event, +100 people registered and +25 talks has been given. The traditional session dedicated to Projects Status allowed some major Tango-based projects to report on their progress. Among them, were SKAO, the world's largest radio telescope under construction and the ESRF-EBS, the first-of-a-kind fourth-generation

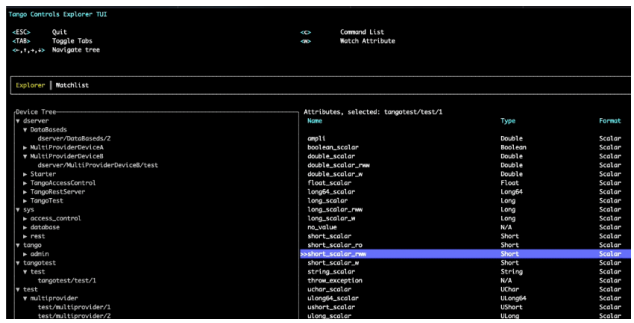


Figure 3: TUI - text based generic client written in Rust.

high-energy synchrotron. The technical sessions related to the Tango ecosystem provided the attendees with latest news of the kernels, the tools and the services around Tango. PyTango - the Python binding based on the C++ kernel - remains the most popular platform for both server and clients development, especially for newcomers. The trend towards the usage of web-based solutions for controls systems GUIs continues, re-enforced by the need for remote controls during the COVID-19 pandemic. The Tango Community Meeting was also an good opportunity to introduce the subcontractors who were selected during the Call for Tender managed by the ESRF.

In addition to regular Tango Kernel Meetings, Tango Kernel Training Sessions and the yearly Tango Community Meeting, the HDB++ Archiving collaboration meetings are now scheduled every two months to boost the collaboration between institutes. These regular follow-ups of the project have enabled the continuous improvement of the main library, the testing and evaluation of the HDB++ core libraries over multiple database engines (PostgreSQL, MySQL, MariaDB, TimeScaleDb, ...) [50] as well as the development of a common AbstractReader class for data extraction.

TANGO V10

Tango V10 is the code name of the next long term version of Tango which should bring a change in the communication protocol. Since the backward compatibility is one of the main attractions for Tango, the community has decided to collectively understand the consequence of this major change. The first step was to define what is the essence of Tango in the form of RFCs.

The Tango Request For Comment RFC project [51] specifies the data model of Tango, the expected behaviour of the Tango elements and the communication between 2

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

components, as it is today with version 9. The project of writing these specifications has involved all representatives of the Tango consortium in such a way that the knowledge is built collectively. The draft of the specifications is almost complete before being compared to the existing implementation. At the current state, it covers the essentials of the Tango Model with 12 specifications.

The main communication protocol is the most complex to specify as a large part of it is given by the CORBA implementation. A careful review of the cppTango library allows extracting the main principle, although separate the abstraction from the implementation is a difficult challenge.

The RFCs are published on ReadTheDoc (see [52]). These documents have status *draft*.

The final work in progress concerns RFC-10/The Request-Reply Protocol. The RFC-13/Publisher-Subscriber protocol implementation with ZMQ is waiting for review. Three other RFCs are planned to be written if needed.

The Tango Community plans to organise a workshop for the final edition of the RFCs and plan the next steps.

The Community will follow COSS and use the RFCs as a base for building a prototype of Tango V10. This will verify the specification for completeness and any ambiguity. When a new version of Tango Controls is ready, the documents will be marked as stable [53].

TRACING VIA LOGS

The SKA project developed a Python library [54] that generates logs with tracing information. A log entry is created upon the entry and exit of the context manager.

The main items in the log messages are:

transaction ID Tango device command arguments are generally type string (JSON). The JSON may include a common transaction ID that is passed along as part of the arguments. By searching for this ID in the logs, the execution path can be traced between devices.

Enter|Exit name Whether this is the start or end of the context, and an arbitrary name for the log pair.

marker A random string used to match "Enter" and "Exit" log entries.

For more details see paper [55]

CONCLUSION

The Tango Controls Collaboration has again proved to essential in ensuring the continued maintenance and development of the Tango kernel for the coming years and decades thanks to securing high quality sub-contractors on multi-annual contracts. The high quality of the in-kind contributions continue to contribute to the successful development of Tango Controls. Major new projects are based on Tango which puts a strong requirement on sustainability for the coming decades. A new binding is now available for Rust. Rust is a promising language for the future which addresses the problems of memory leaks which traditional languages like C++, C and Python suffer from. The web developments have been pushed ahead with the wide-spread adoption of the Tango GraphQL protocol. The next step will be to fix the specification of Tango GraphQL to ensure compatibility between multiple implementations. The collaboration is setup for the next 5 years and is open to new members who rely on Tango or plan to use Tango for their facilities and products.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Tango Controls Collaboration for funding a number of the developments described here as well as the Tango Controls Community for bug reports, fixes, suggestions for new features and contributions.

REFERENCES

[1] T. Juerges, J. Mol, T. Snijder "LOFAR2.0: Station Control Upgrade", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOAR03, this conference.

[2] A. Trifonov, M. Gostkin, V. Kobets, M. Nozdrin, A. Zhemchugov, P. Zhuravlyov "The Control System of the Linac-200 Electron Accelerator at JINR", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUAR03, this conference.

[3] L. Li, J. Luo, Z. Ni "Fast Creation of Control and Monitor Graphical User Interface for Pepc of Laser Fusion Facility Based on Icsff", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper THPV007, this conference.

[4] S.M. White *et al.*, "Commissioning and Restart of ESRF-EBS", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 1–6. doi:10.18429/JACoW-IPAC2021-MOXA01

[5] https://indico.cells.es/event/619/contributions/1480/attachments/1034/1691/ESRF-EBS-Status-Tango-Community-Meeting-Sep.2021.pptx

[6] J. Santander-Vela, M. Bartolini, M. Miccolis, N. Rees "From SKA to SKAO: Early Progress in the SKAO Construction", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOAL03, this conference.

[7] https://github.com/tango-controls

[8] https://travis-ci.org

[9] M. Liszcz et al "Migration of Tango Controls Source Code Repositories", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOPV034, this conference.

[10] https://github.com/tango-controls/cppTango

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

[11] https://github.com/tango-controls/cppTango/releases/tag/9.3.4

[12] https://github.com/tango-controls/TangoSourceDistribution/releases/tag/9.3.4

[13] https://gitlab.com/tango-controls/cppTango/-/blob/9.3.4/CHANGELOG.md

[14] https://clang.llvm.org/docs/ClangStaticAnalyzer.html

[15] https://clang.llvm.org/extra/clang-tidy

[16] https://clang.llvm.org/docs/AddressSanitizer.html

[17] https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html

[18] https://clang.llvm.org/docs/ThreadSanitizer.html

[19] https://gitlab.com/tango-controls/pytango/-/releases/v9.3.3

[20] A. Götz, G. Abeillé, M. Bartolini, R. Bourtembourg, T. Braun, J.M. Chaize, *et al.*, "State of the Tango Controls Kernel Development in 2019", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1234–1239. doi:10.18429/JACoW-ICALEPCS2019-WEPHA058

[21] https://www.boost.org/doc/libs/1_76_0/libs/python/doc/html/index.html

[22] https://github.com/pybind/pybind11

[23] https://www.jacorb.org/.

[24] https://github.com/zeromq/jeromq

[25] https://bintray.com/.

[26] https://search.maven.org/artifact/org.tango-controls/JTangoServer

[27] https://jfrog.com/blog/into-the-sunset-bintray-jcenter-gocenter-and-chartcenter/.

[28] https://tango-controls.readthedocs.io/en/latest/tools-and-extensions/built-in/pogo/index.html

[29] https://gitlab.com/tango-controls/pogo

[30] https://mvnrepository.com/artifact/org.tango.tools.pogo.gui/Pogo

[31] http://pubrepo.maxiv.lu.se/rpm/el7/x86_64/.

[32] https://gitlab.com/tango-controls/tango-spec

[33] https://copr.fedorainfracloud.org/coprs/g/tango-controls/tango/.

[34] https://docs.conda.io

[35] https://anaconda.org/tango-controls

[36] The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem, doi:10.5281/zenodo.4774216

[37] https://github.com/conda-forge/conda-forge-pinning-feedstock

[38] M. Canzari *et al.*, "Satisfying wishes for SKA engineers: how Taranta suite meets users' needs", in *Software and Cyberinfrastructure for Astronomy VI*, SPIE, 2020, pp. 700–706. doi:10.1117/12.2562585

[39] https://gitlab.com/tango-controls/rest-api

[40] J-L.Pons, "TangoGraphQL: A GraphQL binding for Tango control system Web-based applications", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOPV025, this conference.

[41] https://indico.cells.es/event/619/contributions/1462/attachments/1044/1728/Vue_TangoGQL_SOLARIS.pdf

[42] https://vuejs.org/.

[43] https://gitlab.com/tango-controls/web/tangogql

[44] https://indico.cells.es/event/619/contributions/1463/attachments/1039/1708/IC@CMSStatusTangoCommunityMeeting.pdf

[45] https://www.rust-lang.org/.

[46] Jeffrey M. Perkel, "Why scientists are turning to Rust", *Nature*, vol. 588, pp. 185–186, 2020. doi:10.1038/d41586-020-03382-2

[47] https://gitlab.com/tango-controls/tango-rs

[48] https://crates.io/crates/tango-client

[49] https://github.com/SKAJohanVenter/tango-controls-tui

[50] R. Bourtembourg, G. Cuní, M. Di Carlo, G.A. Fatkin, S. James, L. Pivetta, *et al.*, "Pushing the Limits of Tango Archiving System using PostgreSQL and Time Series Databases", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1116–1121. doi:10.18429/JACoW-ICALEPCS2019-WEPHA020

[51] https://gitlab.com/tango-controls/rfc

[52] https://tango-controls.readthedocs.io/projects/rfc/en/latest/.

[53] P.P. Goryl, V. Hardion *et al.*, "Tango Controls RFCs", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUBL03, this conference.

[54] https://gitlab.com/ska-telescope/ska-ser-log-transactions

[55] S.N. Twum *et al.*, "Implementing an Event Tracing Solution With Consistently Formatted Logs for the SKA Telescope Control System", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUBL02, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR02

ADAPTATIONS TO COVID-19: HOW WORKING REMOTELY HAS MADE TEAMS WORK EFFICIENTLY TOGETHER

R. Lacuata, B. Blackwell, G. Brunton, M. Fedorov, M. Flegel, D. Koning, P. Koning,
S. Townsend, J. Wang, Lawrence Livermore National Laboratory, Livermore, CA , USA

Abstract

The National Ignition Facility (NIF) is the world's largest 192 laser beam system for Inertial Confinement Fusion (ICF) and High Energy Density Physics (HEDP) experiments. The NIF's Integrated Computer Control System (ICCS) team conducts quarterly software releases, with two to three patches in between. Each of these software releases consists of deployment, regression testing, and a test shot. All of these are done with ICCS software team members inside the NIF control room. In addition, the NIF ICCS database team also performs a dry run and verification before each software release. This is to anticipate any issue that may arise on the day of the release, prepare a solution for it, and make sure that the database part of the upgrade will be completed within the allotted time slot. This paper describes how the NIF ICCS software teams adapted when the LLNL workforce began working remotely due to the COVID-19 pandemic. These adaptations led to a better and more efficient way of conducting the NIF ICCS software upgrades.

INTRODUCTION

I joined NIF ICCS back in October of 2017. My background was database development and support for an e-commerce company. A few weeks later, I participated in my first NIF ICCS software release. This is when I realized that updating the control systems for the NIF is different than the software releases I did in my previous job.

The NIF ICCS software releases are done quarterly, with two to three patches in between. It requires a lot of effort from operations, hardware, and software teams.

The software release process is pretty standard. It consists of a database dry run and verification, deployment, regression testing, and a test shot. The big difference from my previous job's software release is with the test procedures done to requalify the changes. The qualification testing in NIF ICCS is more involved that just testing the new version of the application with a web browser.

The pre-pandemic version of our software release process involves a lot of group activities. Most of which requires the software team members to go to the control room to execute the task they needed to do.

For example:

Before the release, we perform the database release dry run. This is done in a team member's office. After a successful dry run, we will go to the control room to verify that the database release scripts have been delivered and the console we will use for deployment is working. This is because the production database is accessible only from the control room.

On the day of the release, we will be in the control room again to apply the database updates. After we are done, members of operational staff will activate the new software that was delivered by our configuration management team.

Once the new software is activated, it will become busy in the control room. Members of the software team will start to come in to do regression testing (Fig. 1). Since only members of the operational staff are qualified to operate the laser, they will have to pair up with an operational staff member to run a series of test procedures for them. The operational staff would sit at their consoles and perform the required operations while they stand behind and observe software behaviour.

Elsewhere in the facility, engineers from other teams will be executing their part of the software release to upgrade various computers, servers, and controls hardware.

All of these release activities and qualification testing are coordinated by the release manager who is also inside the control room. He gets constant updates everybody.

Figure 1: Pre-pandemic, ICCS software engineers inside the control room watching the operators' consoles [1].

When the pandemic started, Lawrence Livermore National Laboratory (LLNL) went into Minimum Safe Operations in March 2020. Only essential personnel are allowed onsite. Most of the laboratory's workforce started working remotely, including the ICCS software team.

Because of this, the software release that was scheduled in April 2020 was cancelled.

After some modifications to control room operations to comply with COVID-19 restrictions, NIF shot operations was able to resume. This required the software team to resume software releases to the NIF control systems as well. But the software release process that required most of the software team inside the control room is not possi-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR02

ble. COVID-19 restrictions forced us to rethink our software release process.

NIF ICCS management team met with the software team to figure out how to execute a software release while the entire software team is working remotely. The software release procedure was updated. The changes that were meant to address COVID-19 restrictions also led to a better and more efficient way of conducting software releases.

LEVERAGING CLOUD COLLABORATION PLATFORMS

We have used most of these tools before the pandemic. For one reason or another, we just have not considered using them for software releases.

Virtual Desktops

The software team is able to continue development and testing activities remotely because they can connect to the development and testing environments. To support production releases, the NIF provided the database team with virtual desktops. The virtual desktop opened secure connections to computers, databases, and servers in the production environment. This replaced the need for the database team to be in the control room to do database updates and verification. With the database team and the rest of software team working on their own workstations, they have access to their computers and developer tools that would not have been available if they were in the control room. This is invaluable to any software engineer. This has made the software team more effective during the release.

Smart Phones and Cisco Webex

The NIF issued smartphones to each operational staff member. The software team also leveraged an established video conferencing capability using Cisco Webex to view any operational console inside the control room. The combination of these two provided secure one-to-one audio/video communication between the operational staff and the software team. This replaced the need for the software team to be in the control room during regression testing.

The release manager does not have to be in the control room as well. He too is working from his own workstation. He schedules a single Webex meeting for the entire duration of the software release to coordinate all activities and qualification testing.

When issues come up, remote and onsite teams simply join this online meeting to collaborate and solve problems. Team availability during software releases improved.

Microsoft Teams

Microsoft Teams provided a platform for the teams to collaborate online. It brings the operations, hardware, and software teams in one place where they can work together, share information, and make decisions to drive the software release into successful completion.

The release manager establishes a Microsoft Teams channel dedicated for each software release where the release activities and qualification testing are organized as channel conversations.

The conversations and status updates from the release channel replaced the face-to-face conversations that used to take place in the control room and status updates via phone calls and e-mails.

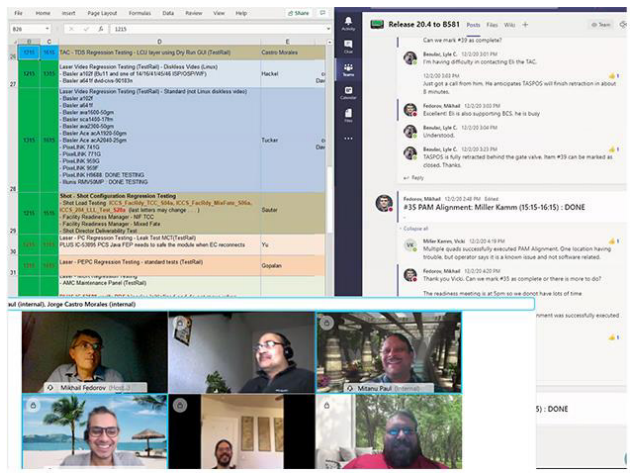


Figure 2: The ICCS software team uses Cisco Webex and Microsoft Teams during one of the software releases [1].

CONCLUSION

The remote software release met COVID-19 social distancing requirements and more.

For the software team that used to crowd the control room on release days, they can now work comfortably in their own offices, whether onsite or remotely at home (Fig. 2).

For the operational staff working in the control room, the reduced number of people made it more a comfortable working environment for them.

Remote software release facilitated online collaboration. The release manager shares the overall status of the release online. This combined with the status updates from the release channel increased team awareness and communication. If there are issues that needs attention, the team response is much faster. Escalations and resolutions also are recorded in the release channel, makes data available for later analysis.

Remote software release also provided an opportunity to train new members. The new member joins the video conference and watch as the mentor executes the release procedures. Before, this meant bringing the new member in the control room which is not always possible. Now, we have more opportunities to train them.

The software team has completed six remote software releases since the pandemic began. It is such an improvement from the old process that the software team will continue with this new process onwards.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-827816.

REFERENCES

[1] P. Koning, "Remote NIF Control Software Updates Lead to Less Stress, More Efficiency", NIF & Photon Science News, February 11, 2021.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR03

AGILITY IN MANAGING EXPERIMENT CONTROL SOFTWARE SYSTEMS

K. V. L. Baker, F. A. Akeroyd, T. Löhnert, D. Oram, ISIS Neutron and Muon Source, Didcot, UK

Abstract

Most software development teams are proponents of Agile methodologies. Control system software teams, working at science facilities, are not always just devel-opers, they undertake operations work, and may also be responsible for infrastructure from computer hardware to networks.

Parts of the workflow this team interacts with may be Agile, but others may not be, and they may enforce deadlines that do not align with the typical agile implementations. There is the need to be more reactive when the facility is operating, which will impact any development work plans. Similarly, friction can occur between an Agile approach and more familiar existing long-standing risk-averse organisational approaches used on hardware projects.

Based on experiences gained during the development of IBEX, the experiment control software used at the ISIS Pulsed Neutron and Muon source, this paper will aim to explore what being Agile means, what challenges a multi-functional team can experience, and some solutions we have employed.

WHAT DOES AGILE MEAN?

In its' truest form Agile is a way of working [1] rather than the tools to enable this way of working. What is most important to remember is that the manifesto values certain things over others, but the less valued items are still worth considering, just not at the expense of the more valued items. The Agile Manifesto values are as follows:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

It was software developers that started the use of Agile methodologies, as it is often in the fast-paced world of software that the adaptability is most important in the modern age. However, some of the tools used to support Agile methodologies originated separately from software development, and Agile Project Management is slightly different to Agile Software Development.

What is Agile Project Management?

The Association for Project Management [2] describe Agile Project Management as "an iterative approach to delivering a project throughout its life cycle" [3].

The Agile Manifesto and Principles are generally applied in exactly the same way whether the project in question is software based or not, and instead of working software at each iteration, you aim for working prototypes or solutions.

Is Agile Always the Right Answer?

It is certainly true that not every project is suited to using an agile methodology. Yet, there are very few project teams who would deny that continuous collaboration throughout the project from the customer or their representative, and being able to incorporate any requested, or required, changes over the course of a long term project is beneficial to producing a usable item at the end. Some of the tools which are typically seen in Agile, have other roots. A common tool used to map workflows is a board such as a Kanban board (see Fig. 1) [4].

Initially the use of Kanban was by the assembly lines at Toyota [5], but the tool is used by most Agile methodologies to track progress.

Whilst Agile isn't the answer to everything, aspects of it, and the tool sets it uses are still applicable outside of Agile projects, and vice versa. If using any of the standard toolsets used by Agile Project Management, it is worth making sure you use the one that most suits your team and environment, and to continuously evaluate the tools suitability for use as the project and team develops.

MULTI-FUNCTIONAL TEAMS

Whilst the ideal can be to have teams focussed on just one thing, the practicalities mean that people regularly undertake a variety of tasks.

Types of Function

For the purposes of this paper a function is defined a family of tasks that an individual can undertake.

Development (Dev) The obvious task undertaken by most software developers is development. This mainly covers the introduction of new features to a code base, for example, adding in code to allow a user to change the colour of the interface they are using.

Development covers the full software stack, from the user interface to the lowest levels the team can support, which may even be circuitry. For non-software teams this could be electronic systems, or mechanical ones.

Systems (Sys) The focus for systems tasks is usually the hardware and fundamental aspects of the environment the control system is running in, e.g. computers, network switches, and operating systems. It also covers patching and updating the elements mentioned.

Operations (Ops) Operations tasks are responses to requests for support on problems that need to be solved in a short time frame to keep systems and software running correctly.

It also covers some of the later parts of the process, such as fixing code. For software teams this is hearing about bugs and dealing with them. It is more likely to be failures and faults for those who are not focussed on software, and is the nature of operations for the Systems function. As such

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

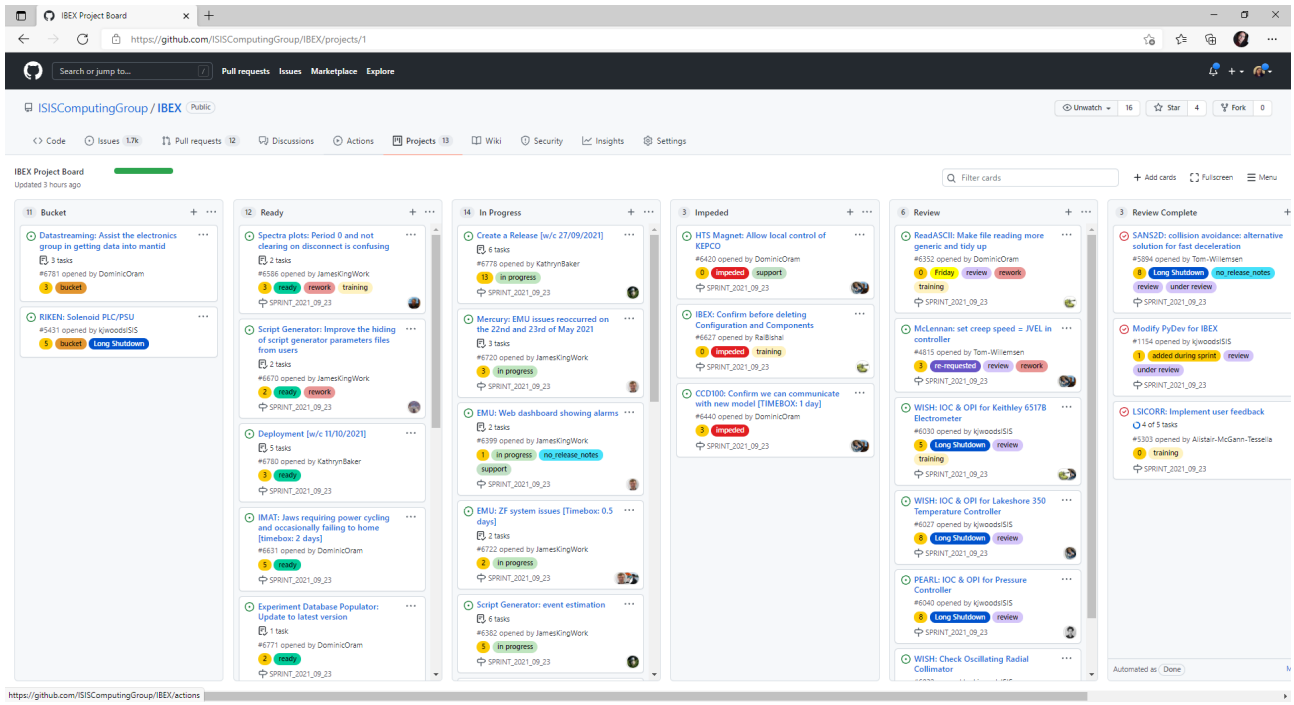


Figure 1: An example of a Kanban board, the one in use on the IBEX project at the start of October 2021.

operations teams can need an understanding of what both the systems and development work entails, although not always to the same depth.

Project Management, Admin Whilst it can be the case that project management and admin tasks are undertaken by specialists in those fields. It isn't always the case, and it can fall to the same team that is designing the software to control an experiment to manage the project and to keep the admin tasks related to that system up to date.

This can also be stakeholder management, ensuring that those communication and collaboration channels are kept open for dealing with our scientists.

What the Basic Challenges are for Multi-Functional Teams

If you are able to split the work between team members with defined functions that do not overlap, then whilst the team might be multi-functional, the only challenges occur when those within a function are not available, and that can be the case for any specialist aspect of the control system.

For a team where each member can be involved in multiple functions, the challenges are often quite different. Not least of which is context switching. If a team member is undertaking some programming work, and is interrupted by a phone call reporting a failed network connection, there is a moment where they have to switch between thinking in a programming language, to how to go about troubleshooting what the issue with the network might be. This kind of switching between tasks has a cost [6], which can have a significant impact on the productivity of a team.

Similar to the context switching, if people are working on multiple projects then interruptions will have an effect on productivity too [7]. The issue can come on how you differentiate between development work and operational work, and just how many projects the team/individuals are working on.

It is also rare that a team member can be specialist enough in all those areas to be able to provide the same level of support, which can be frustrating for those seeking help of an operations function.

Where Agile can Help

Due to the central precept of embracing change Agile methodologies allow for systems to change frequently, which will be of benefit to the ever changing needs of a science facility.

If you have individuals with specialist skills then it can be easier to manage their input across multiple projects.

Where Agile Hinders or is Counterintuitive

The thing we have found causing the greatest friction for us relates to how things are defined as complete. Our iterative approach to development provides a Minimum Viable Product (MVP) and then adds to the product, which often leads to difficulty in defining when a product is in a finished state, effecting team morale.

With a facility such as ISIS, where the customer representatives (the scientists who take care of the instrument) work extended hours during the times that the accelerator is running it can be hard to find their time, or for them to prioritise the collaboration aspects needed for Agile working. Especially when combined with the iteration process.

For a multifunctional team however, Agile methodologies can increase context switching, as the next most important

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR03

thing to do might be related to something completely different to what you were just working on, but or the developer can find themselves jumping from one programming language to another.

HOW OUR TEAM FUNCTIONS, AND HOW WE ARE AGILE

What Type of Team is the IBEX Team?

Within the realm of supporting control software on instruments at a facility such as ISIS [8], we cover all functions within the team. Some members of the team focus more on the Systems side, others on development, and still others will undertake the management tasks.

All members of the team take on an operations role, especially when the ISIS accelerator is running, as that is when the instruments are most active. As the available beam time is limited, we work to support the instruments to ensure that the time can be used to the best possible advantage.

The whole team is also encouraged to take on responsibilities for different parts of the system as their skill sets grow and improve, so that they act as a point of contact to a beamline.

The IBEX team could be described as a SysDevOps Team, and we cover the management tasks as well.

How the IBEX Team Employs Agile Methodologies

At present we use a modified Scrum approach.

Scrum [9] is one of many Agile frameworks. It provides a heuristic way of tracking the work that has been done, and aids in predicting what can be done. It is focused on the team being self-organising. A Scrum team should consist of three main roles:

- A Scrum Master [10] who looks after the Scrum processes and focuses on keeping the work flowing easily by removing impediments to the work
- A Product Owner [11] who ensures that the product being produced is what is actually useful and who makes sure the priorities are suitable understood
- The Developer role [12] as well as actually developing the product being produced are heavily involved in the planning and defining the end point of the work.

Scrum teams work in timebox [13], which for Scrum are referred to as Sprints. A Sprint is a predefined period of time inside which you do as much work as you can and then consider what has been achieved. Knowing what you have achieved in the past allows you to predict what you can achieve next. These predictions help manage expectations as to what will be provided when, with the feedback of what has been achieved before.

Each Sprint starts with a planning meeting, where the work to be produced is agreed. How much and which items are influenced by the availability of the developers, and the items that the product owner has identified as most important to achieve. The requirements for each package of work should also be finalised here.

Each day in the Sprint the team, including the Scrum Master and Product Owner, meet to discuss what is being worked on and what issues may have been found. Identifying those issues, or impediments if a Developer is waiting on information from a third party, in good time allows the others in the team to help as appropriate.

At the end of each Sprint the work produced is demonstrated to as many people as are interested, users and Scrum team members.

The team also puts time aside after the demonstration to discuss in a retrospective the Sprint, what went well, what went badly, what could be done differently. Over time those retrospectives will shape the way the team works potentially to improve the workflows and interactions based on the current team members and the wider work environment of the team.

How We have Adapted Scrum to Work for us

When we implemented our Scrum system we found one big barrier to following the standard framework, it was hard to identify a single Product Owner who could give us the time needed to take on that role. Instead, the developer team took on an element of that, with individual developers talking to groups of users and refining requirements for what those users were interested in developing. Whether that was a specific ISIS Instrument being converted from our older control system to our new one, or functionality like the script generator [14] which runs across many of the instruments. The priorities are often set in a less detailed way by a Project Board which is concerned about the running of the project, and the Science Advisory Group which is concerned with looking for things that impact and improve multiple instruments or scientific disciplines. This disconnect can lead to tension within the planning meetings as there isn't a single voice dictating the priority, and often leads to many of the tasks not being achieved in the best possible order. It also increases the amount of time needed to manage the expectations of our users.

Similarly, it was hard to gather any of our users to demonstrations at the end of each 20 day Sprint, certainly when ISIS was running their time was not available to us. As such, our demonstrations are made just to the development team each Sprint. When we deploy the latest version of IBEX to instruments we talk to the team of scientists who run instruments in appropriate batches to demonstrate what has changed since we last spoke to them, which can be many Sprints worth of work.

Due to the inclusion of operational tasks as well as development ones our stand up migrated from just an update, to a check on services and systems to ensure we got to errors before they occurred as well as the update.

Those operational tasks also mean that we have implemented a pre-planning meeting, to undertake an initial view of the work to be done in the upcoming Sprint, so that we are not spending the whole day in a planning meeting. This also benefits the focus and well-being of the team.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Where Scrum hasn't been the Answer for us

The rigid timeboxes of Scrum, which are usually the same length, have often been unhelpful for the team. Attempting to undertake the change from one Sprint to the next during heavy operational times, some of which can be predicted, added a layer of stress to the process.

Whilst not specific to Scrum, Agile software development tends to employ a code and refactor mentality. You code the bare minimum to achieve what is required, and as you add new functionality you refactor the code to keep it looking tidy. The issue is the larger your code base the harder this can be, and as the development of IBEX has proceeded the harder it can be to implement a change for certain instruments without a large refactoring process which reduces the throughput of the team, and the team starts to lose any of the gains made with the other Agile processes compared to a more traditional project management framework.

Typically, Science focused organisations prefer those more traditional methods, as the code and refactor method is not easy to perceive, especially if you are not involved directly in the team and the work being done.

Where Scrum has Worked Well for us

The use of a single backlog, and individual developers picking up the next ticket in the list has ensured that the skills and knowledge of the system has been shared well across the whole team has been beneficial for the operations side of our work.

The regular consideration of whether our way of working is appropriate has been beneficial as well. This has allowed our processes to grow and adapt to support the ISIS instruments in the best way.

What the IBEX Team are Currently Trying

In keeping with those continuous changes as highlighted from the retrospectives, the team has been looking at other ways of working.

Because of the nature of deploying and dealing with an accelerator that runs in cycles of supplying beam to the instruments we are also experimenting with variable length sprints. Varying the duration of the sprint means that we can avoid the sprint end/start meetings falling when we anticipate that operational requirements will be high.

At present we are trying a different way of filling our list of tasks to undertake in a Sprint. Previously we would look at all the items that have to be worked on, which could mean that important items that didn't have a forceful enough advocate in the planning meeting would be missed. Instead, we are assigning certain members of the team to a "theme" which is more in keeping with eXtreme Programming (XP) [15], which is a different Agile framework.

XP is aimed at solving some of the issues found in large and brittle codebases that have been adjusted to fulfil ever-changing requirements. Adding some of the other XP practices as a team may improve the IBEX codebase, especially the even greater emphasis on testing and the encouragement to have the programmer consider the actions of the user.

What We might Try Next

Given that the prioritisation is a recognised stalling point, and whilst the team is small the number of products supported are numerous, some of the concepts employed by SAFe [16], which is the Scaled Agile Framework, might benefit the management of the workload. For example, having a "Program Backlog" for a longer term timebox, 3 months or longer, which agrees the work to be undertaken by the team from the business perspective. The team can then plan Sprints with a Scrum flavour, or XP, or more traditional methods, to achieve those agreed items.

CONCLUSION

Whatever framework is used, Agile or non-Agile, the one tool that does seem to live up to the hype is the Kanban board. As mentioned previously it came from the assembly lines at Toyota, and the ability to track a task or item though an interface that is visible to all can lead to a sense of accomplishment. Whichever aspect of our work is being considered, whether it is an operational task or a new feature, being able to see whether it is waiting to be started, is being worked on, or is complete at a glance is one of the most useful actions we started using.

Different Agile frameworks, such as Scrum, XP, or SAFe, all have benefits and issues. Whilst pure Scrum would not suit the kind of rhythm we have at ISIS in relation to when user cycles start, it can be modified. Other frameworks may offer tools or ways of thinking to overcome the issues we have. However, Agile is a methodology, and the core concepts and where to place value are independent of any framework, and worth bearing in mind when undertaking any work.

REFERENCES

[1] https://agilemanifesto.org/

[2] https://www.apm.org.uk/

[3] https://www.apm.org.uk/resources/find-a-resource/agile-project-management/

[4] https://dictionary.cambridge.org/dictionary/english/Kanban

[5] https://mag.toyota.co.uk/kanban-toyota-production-system/

[6] https://www.apa.org/research/action/multitask

[7] https://www.researchgate.net/publication/317989659_Impact_of_task_switching_and_work_interruptions_on_software_development_processes

[8] https://www.isis.stfc.ac.uk/Pages/home.aspx

[9] https://www.scrum.org/

[10] https://www.scrum.org/resources/what-is-a-scrum-master

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR03

[11] https://www.scrum.org/resources/what-is-a-product-owner

[12] https://www.scrum.org/resources/what-is-a-scrum-developer

[13] https://www.agilealliance.org/glossary/timebox/

[14] J. C. King *et al.*, "The IBEX Script Generator", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper TUPV049, this conference.

[15] http://www.extremeprogramming.org/

[16] https://www.scaledagileframework.com/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

FAIRMAT - A CONSORTIUM OF THE GERMAN RESEARCH-DATA INFRASTRUCTURE (NFDI)

H. Junkes*, P. Oppermann, R. Schlögl, A. Trunschke, Fritz Haber Institute, Berlin, Germany
M. Krieger, H. Weber, Universität Erlangen-Nürnberg, Erlangen, Germany

Abstract

The FAIRmat project, was selected on Friday, July 2, 2021 by the German Joint Science Conference (Gemeinsame Wissenschaftskonferenz – GWK) in a multi-stage competition of the National Research Data Infrastructure (NFDI). The project will receive funding to set up an infrastructure that helps making materials–science data FAIR: Findable, Accessible, Interoperable, and Reusable. This will enable researchers in Germany and beyond to store, share, find, and analyze data over the long term. During the five-year term, a total of 60 project leaders from 34 German institutions will work together in the FAIRmat consortium[1].

FAIRMAT CONSORTIUM

When applying for funding for the FAIRmat project, the following Objectives, work program and research environment were described in the letter of intent:

> Prosperity and lifestyle of our society are very much governed by achievements of condensed-matter physics, chemistry, and materials science, because new products for the energy, environment, health, mobility, and IT sectors, for example, largely rely on improved or even novel materials. The enormous amounts of research data produced every day in this field, therefore, represent the treasure trove of the 21st century. This treasure trove is, however, of little value, if these data are not comprehensively characterized and made available. How can we refine this feedstock, in other words, turn data into knowledge and value? For this, a FAIR data infrastructure is a must.
>
> Here is where FAIRmat ("FAIR Data Infrastructure for Condensed-Matter Physics and the Chemical Physics of Solids") comes in. By building a FAIR research-data infrastructure for the noted fields, the consortium will lift the treasure trove of materials data, and therewith contribute to a disruptive change in the way science and R&D are conducted. Within FAIRmat the acronym FAIR is interpreted in a forward-looking way: Research data should be Findable and Artificial-Intelligence Ready. This new perspective will advance scientific culture and practice. This will not replace scientists, but scientists who use such FAIR infrastructure may replace those who don't.

Projekt Plan

FAIRmat will install a FAIR [1] data infrastructure for the wider area of condensed-matter physics and the chemical physics of solids. This represents a very broad range of different communities that can be characterized by either different classes of condensed matter (e.g. semiconductors, metals and alloys, soft and biological matter, etc.), by different techniques (e.g. ranging from crystal-growth and synthesis to experimental and theoretical characterization by a multitude of probes), or by functionality (exemplified here by battery materials, optoelectronics, catalysts, etc.). As a consequence, the data produced by the community are enormously heterogeneous and diverse in terms of the 4V of Big Data

- Volume (amount of data)
- Variety (heterogeneity of form and meaning of data)
- Velocity (rate at which data may change or new data arrive)
- Veracity (uncertainty of data quality).

Also note that many research data produced today may appear irrelevant in the context they have been produced. Being regarded as *waste*, they are not published. However, they may turn out highly valuable for other purposes.

So, the R in FAIR (reusability) also means "store, share, and recycle the waste!" To cope with all the diversity and complexity, a bottom-up approach that satisfies the needs of the different subcommunities is a must to foster acceptance by the community and participation of a large number of individual researchers and laboratories. FAIRmat sets out to tackle this challenge by a user-driven approach to develop easy-to-use tools and an infrastructure towards FAIR data processing, storage, curation, sharing, and future use of materials data. For the latter, a major goal of FAIRmat is making data artificial-intelligence (AI) ready.

Data obtained by a certain experimental technique for a specific sample of a selected material are only worth keeping if the sample is fully characterized and apparatus and measurement conditions as well as the measured quantity are described in detail. Likewise, computed data are only meaningful when method, approximations, code and

* junkes@fhi.mpg.de
[1] FAIRmat press release

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

code version, as well as all computational parameters are known. In essence, we need an extensive annotation, i.e. a systematic *metadata* catalogue, also covering ontologies. This also includes the description of provenance and data quality (their usefulness for a given context).

Figure 1: FAIRmat task areas.

To address all these aspects and to support basic science and individual researchers, we have identified several task areas (in short called Areas, see Fig. 1) that are sketched in the following:

- Area A – Synthesis – is dedicated to the full characterization of samples and the corresponding synthesis and growth processes. Without this information, reproducibility of materials and their properties with given quality will be hampered. The specific tasks will consider various synthesis routes, i.e., from the gas, liquid, and solid phases, and by assembly.
- Area B – Experiment – covers the microscopic characterization of materials by a broad variety of measurement techniques. Each of them comes with specific challenges concerning processing, curation, and storage, owing to differences in volume, velocity, data formats, etc. FAIRmat will exemplify its approach in a first phase by a representative selection of experimental probes.
- Area C – Theory and Computation – deals with numerical techniques to compute materials properties. Such techniques differ in the theoretical concepts, approximations, and numerical recipes, depending on the employed software. Our general approach to tackle this diversity relies on the concepts developed by the NOMAD Laboratory [2]. The tasks of Area C concern extensions towards excitations and strongly correlated materials, as well as towards classical (particle-based) simulations and multi-scale modeling.
- Area D – Digital Infrastructure – will be a common brace to all other areas. Specific tasks will be dedicated to processing and decentral storage, creating a network of data hubs at different locations. The FAIRmat Portal, a web-based graphical user interface, including a Materials Encyclopedia [3], will allow for searching, accessing, and inspecting (meta)data from all over Germany (and worldwide). FAIRmat will develop and provide tools, from processing to post-processing, including analysis by artificial intelligence, and it will provide guidance and advancements of electronic lab-books (ELN), laboratory information management systems (LIMS), etc. Tight collaborations with HPC centers will ensure the embedding into the overall NFDI landscape.
- Area E – Use-Case Demonstrators – will highlight how the tools developed in the above areas will benefit different scientific communities and demonstrate hand-shakes and potential synergies with other consortia. The specific tasks of this area will cover in the first phase use cases on battery materials, heterogeneous catalysis, optoelectronics, spintronics&magnetism, metalorganic frameworks, biological physics applications, and artificial intelligence.
- Area F – User Support, Training&Outreach – will reflect our concept for how to engage with the community, to allow researchers to make use and handle the FAIRmat tools. A variety of workshops and other training opportunities will be offered and support for connecting data hubs to the overall data infrastructure.
- Area G – Administration and Coordination – will deal with all coordination and management issues and the embedding into the Overall NFDI. As such, it will address synergies with other consortia and the interaction with the NFDI Directorate.

FAIRmat represents a broad community of numerous researchers from universities and leading institutions in Germany. It builds on extensive experience with the worldwide biggest data infrastructure in computational materials science, the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Novel Materials Discovery (NOMAD) Laboratory and the association FAIR-DI e.V. FAIRmat aims at covering the full breadth of the Condensed Matter Section of the German Physical Society (DPG) with its 12 divisions, and is further supported by the Chemistry, Physics, and Technology Section of the Max Planck Society, the Bunsen Society for Physical Chemistry, and more. It is fully embedded internationally, e.g., in the Research Data Alliance, the European Open Science Cloud, GO FAIR, etc. and has signed Memoranda of Understanding with leading institutions worldwide, for example NIST (USA), Shanghai University (China), and CSC (Finland). FAIRmat will continue to raise awareness and acceptance of a FAIR research-data infrastructure in Germany, Europe, and beyond. [4, 5]

Task D5

A universal and easy-to-configure software environment for measurement data acquisition and documentation is to be developed in Task D5 of the FAIRmat consortium.

In the field of Applied Physics, measurement setups with numerous specific measurement devices are often required – in each case adapted to the experimental problem. The diversity requires adaptable and easy-to-configure software for experiment control and data acquisition. But it's not just about the raw data. The experiment description, including all settings of the laboratory equipment used, the *metadata*, is also required. Only in this way, the experiment is documented completely and FAIR, and the valuable measurement data can be re-used by other scientists.

A prototype software developed at the chair of the Department of Physics at the Friedrich-Alexander-Universität (FAU) Department for experiment control with uniform and documented data output has been using for years [6]. In the FAIRmat project this successful concept will be put together with the open-source Experimental Physics and Industrial Control System (EPICS) [7].

CONFIGURABLE EXPERIMENTAL CONTROL SYSTEM (CECS)

An existing system developed at the Department of Physics (FAU) has been built based on LabVIEW [8].

This system has proven itself at the department in research and teaching (Fig. 2). However, it does not satisfy the FAIR rules. By using LabVIEW, the system is:

- not open source,
- not operating system independent,
- not platform independent,
- is very poorly scalable.

Because of these missing features, the new CECS will be based on EPICS to control the scientific and measurement equipment [9].

Although EPICS is also suitable for small experiments [10], there is a persistent opinion that it can only be used by experts and larger teams of developers. This may also have to do with the fact that the aspect of data acquisition is rarely taught at universities nowadays. One aspect in the development of the new system is the possibility to introduce the users to the data acquisition step by step and transparently.

The system shall provide an easy to use graphical experiment configuration interface like the LAP Measurement before. We decided to implement this configuration interface in python. The packages from Python for Scientific Computing (Fig. 3) are used.

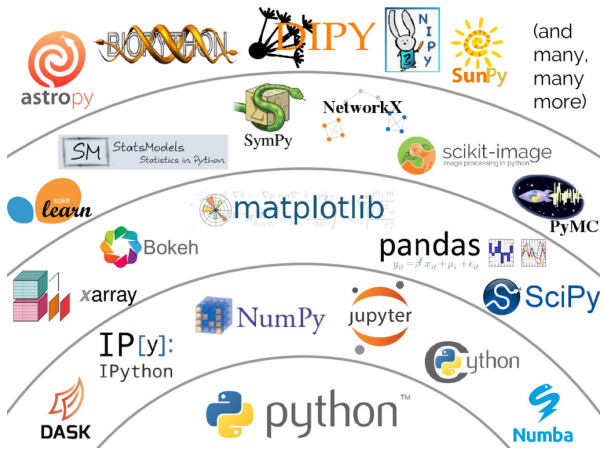


Figure 3: Python for Scientific Computing.

Python has now established itself as a programming language, especially in scientific subjects, and is taught at universities. The premise of CECS is the ability to configure a measurement system graphically without having to have programming knowledge. However, the GUI itself should not yet represent a direct implementation of a measuring program but generate a human (and machine) readable output which then serves as a *recipe* for a measuring program. This gives the possibility to let the configuration system generate a template (python), which one can refine and adapt to special needs. This makes it possible to get deeper and deeper into the system and to adapt it. The experience has shown that the scientist is willing to improve such templates if everything is presented transparently. The configuration system can also be used to specify general structures (e.g. storage location, file format, etc.) and thus achieve a standardized data recording. These *recipes* are excellent for documenting the workflows and can be linked to the documentation system (ELNs, etc). This also makes it possible to save these work steps in a versioning system. This makes it possible to repeat measurements.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

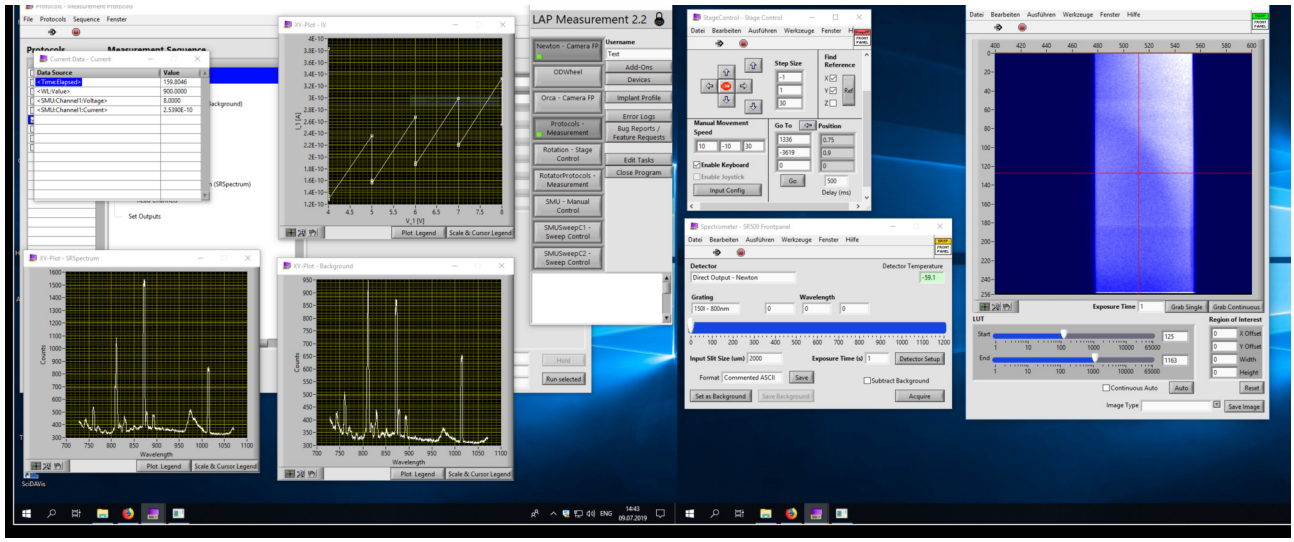


Figure 2: Example of the LAP Measurement User Interface.

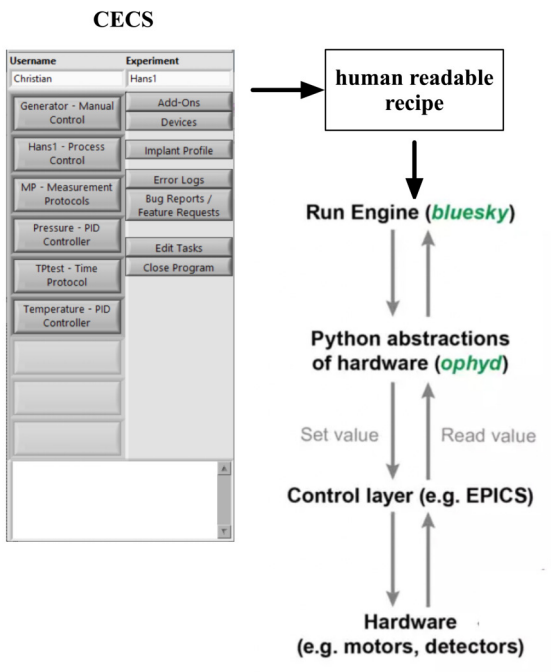


Figure 4: CECS recipe for Bluesky.

With the help of this *recipe*, a plan and virtual devices for Bluesky [11] are created. Bluesky stands on the shoulders of EPICS, and provides additional capabilities such as live visualization and data processing tools, and can export data into nearly any file format in real time. Bluesky was developed using python. That will make Bluesky simple for future scientists to modify, and to implement to new experiments.

EPICS

Complex systems EPICS is a set of software tools and applications which provide a software infrastructure for use in building distributed control systems to operate devices such as Particle Accelerators, Large Experiments and major Telescopes. Such distributed control systems typically comprise tens or even hundreds of computers, networked together to allow communication between them and to provide control and feedback of the various parts of the device from a central control room, or even remotely over the internet.

High bandwidth, soft realtime networking applications EPICS uses Client/Server and Publish/Subscribe techniques to communicate between the various computers. Most servers (called Input/Output Controllers or IOCs) perform real-world I/O and local control tasks, and publish this information to clients using robust, EPICS specific network protocols Channel Access and pvAccess. These protocols are designed for high bandwidth, soft real-time networking applications that EPICS is used for, and is one reason why it can be used to build a control system comprising hundreds of computers.

Flexible and scalable E.g. at the Advanced Photon Source national laboratory in the United States, EPICS is used extensively within the control system for the accelerator (see Fig. 5) and many of the experiments. There are about hundreds of IOCs that directly or indirectly control almost every aspect of the machine operation, while 40 workstations and servers in the control room provide higher level control and operator interfaces to the systems, and perform data logging, archiving and analysis.

Roadmap EPICS is developed through a collaborative open-source process where anyone is free to contribute to the EPICS family of software. In addition to this, heavy EPICS users – typically large scientific facilities – gather together in an open EPICS council to define a roadmap for the future direction of EPICS.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

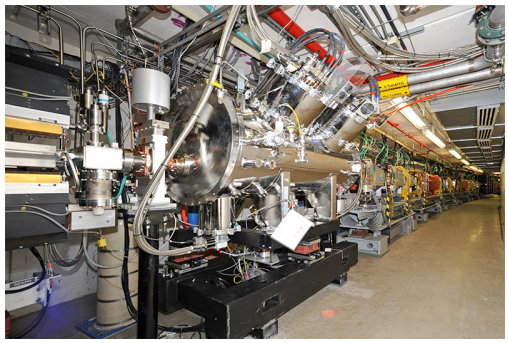


Figure 5: Beampipe at APS.

License EPICS is provided under an open source license called the EPICS Open License, which is similar to the BSD license. The EPICS licensing page [12] gives details and some history.

Devices supported The IOC, the Input/Output Controller is the I/O server component of EPICS. Almost any computing platform that can support EPICS basic components like databases and network communication can be used as an IOC. One example is a regular desktop computer, other examples are systems based on real-time operating systems like vxWorks or RTEMS and running on dedicated modular computing platforms like MicroTCA, VME or CompactPCI. EPICS IOC can also run on low-cost hardware like RaspberryPi or similar.

IOCs can support any number of records and record types. Similarly, record support does not contain device-specific knowledge, so each record type can have any number of independent device support modules. If the method for accessing hardware is more complicated than device support, then a device driver can be developed.

Currently, there are already a number of modules available to support instruments that are also commonly used in many experiments, such as temperature controllers (Eurotherm) and mass flow controllers (Bronkhorst) [liste]. These individual instruments can be connected to the IOC via serial or Ethernet interfaces.

Bluesky

Bluesky [11] is a mini-ecosystem of co-developed but individually useful Python libraries for experiment control and data acquisition, management, and access. The project is developed and maintained by a multi-facility collaboration. The core includes a high-level hardware abstraction above EPICS, an experiment orchestration engine, a formally-defined schema for streaming data and metadata, and data access tools integrated with the open source scientific Python stack (see Fig. 6). It emphasizes the following virtues:

- Live, Streaming Data: Available for inline visualization and processing.
- Rich Metadata: Captured and organized to facilitate reproducibility and searchability.
- Experiment Generality: Seamlessly reuse a procedure on completely different hardware.
- Interruption Recovery: Experiments are *rewindable*, recovering cleanly from interruptions.
- Automated Suspend/Resume: Experiments can be run unattended, automatically suspending and resuming if needed.
- Pluggable I/O: Export data (live) into any desired format or database.
- Customizability: Integrate custom experimental procedures and commands, and get the I/O and interruption features for free.
- Integration with Scientific Python: Interface naturally with numpy and Python scientific stack.

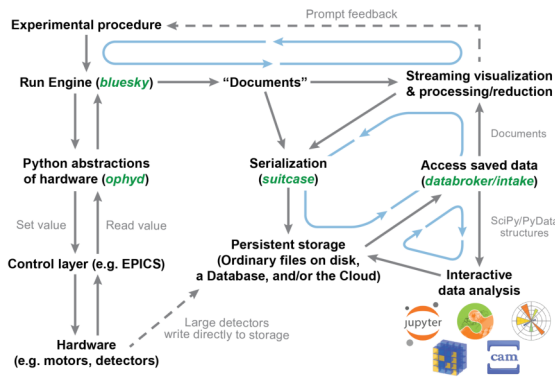


Figure 6: Bluesky project.

APPLICATION AREAS

Another example of the ideal use of such a low-threshold data acquisition system is in catalysts research.

The challenges of the energy transition can only be solved with the help of innovative catalyst technologies. In catalysis research, a number of empirical concepts exist and predictions of theory are based on first-principles-based models that are subject to a variety of assumptions [13]. For simple reactions, material-function relationships have been predicted very successfully in this way. However, this is much more difficult for complex reactions and catalyst systems that are subject to high dynamics. Energy-relevant reactions, such as the hydrogenation of carbon dioxide, or important catalytic conversions for chemical hydrogen storage, are such complex reactions. Here, catalyst dynamics, i.e. the change in the state of the catalyst depending on the history of the catalyst and the reaction conditions, complicates the application of Big Data analyses and data mining methods because experiments performed according to different workflows can lead to inconsistent data sets. On the other hand,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL01

the complex relationships between catalyst structure and functional properties of the catalyst can probably only be decoded with reasonable effort using artificial intelligence methods. For this purpose, reliable, reproducible data sets with high diversity are required. Handbooks in which the characterisation of catalysts and the determination of kinetic data are precisely prescribed serve to generate such data. These handbooks should specify what minimum data set should be generated for each catalyst and how the measurements should be performed. An example of a workflow in catalysis research is shown in Figure Fig. 7.

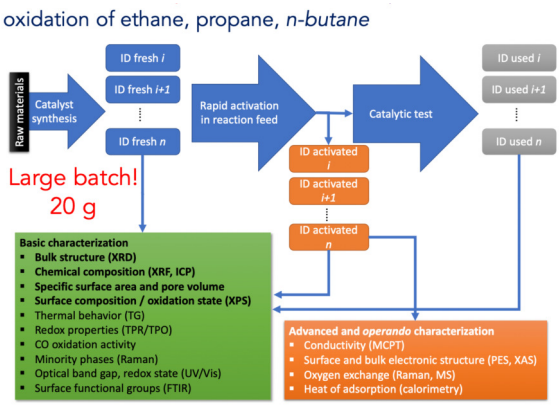


Figure 7: Workflow catalyst investigation.

These handbooks can then be converted into readable (human and machine) formats which can then be e.g. read as *recipes* by Bluesky.

For a typical setup in catalayse research see Fig. 8.

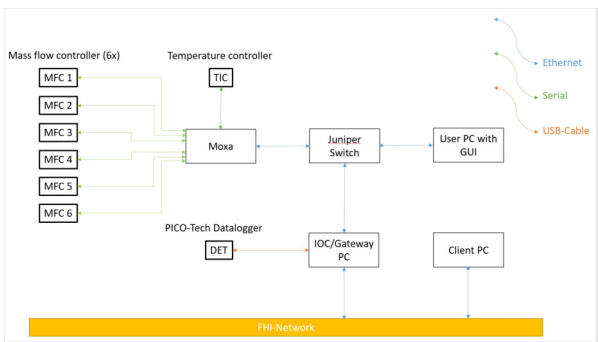


Figure 8: Block diagram of a reactor for testing catalysts.

A gateway computer (see Fig. 9) sets up its own experiment subnet and collects all the data. It is equipped with two Ethernet interfaces and up to 20 serial interfaces. In addition, custom device drivers were written to communicate with analog-to-digital converters, the Modbus protocol is used for this purpose.

ACKNOWLEDGEMENTS

Alexander Fuchs and Johannes Lehmeyer (FAU) are implementing the new CECS system as part of the FAIRmat project. Abdulrhman Moshantaf (FHI) implements the control system for the CatLab reactor systems.

Figure 9: Jetway IOC.

REFERENCES

[1] FAIR principles, https://www.go-fair.org/fair-principles/.

[2] NOMAD-Laboratory, https://nomad-lab.eu/.

[3] NOMAD materials encyclopedia, https://www.nomad-coe.eu/index.php?page=materials-encyclopedia

[4] FAIRmat-LoI, https://www.fair-di.eu/uploads/documents/FAIRmat_LoI_2020_Website.pdf

[5] FAIRmat-Template for Submission of LoI, https://www.fair-di.eu/uploads/documents/FAIRmat_Abstract.pdf

[6] C. Ott "LAP Measurement v2.2: Talk", Private communication, LAP_Measurements_Kaffeeseminar_2019-11-25

[7] EPICS: Collaboration website, about EPICS, https://epics-controls.org/about-epics/.

[8] LabVIEW: Company website, https://www.ni.com

[9] FAIRmat: Lifting the treasure trove of materials data https://www.physics.nat.fau.eu/2021/07/05/fairmat-lifting-the-treasure-trove-of-materials-data

[10] H. Junkes, "EPICS Also for Small and Medium Sized Experiments", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1269–1272. doi:10.18429/JACoW-ICALEPCS2019-WEPHA075

[11] Bluesky: Project description, https://blueskyproject.io/.

[12] EPICS: License information, https://epics-controls.org/licensing/.

[13] A. Trunschke *et al.*, "Towards Experimental Handbooks in Catalysis", *Topics in Catalysis*, vol. 63, pp. 1683–1699, 2020. doi:10.1007/s11244-020-01380-2

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

PROTOTYPE OF IMAGE ACQUISITION AND STORAGE SYSTEM FOR SHINE*

Huihui Lv, Huan Zhao†, Danping Bai, Xiaomin Liu
Shanghai Advanced Research Institute, Chinese Academy of Sciences
201204 Shanghai, P.R. China

Abstract

Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE) is a quasi-continuous wave hard X-ray free electron laser facility, which is currently under construction. The image acquisition and storage system has been designed to handle a large quantity of image data generated by the beam and X-ray diagnostics system, the laser system, etc. A prototype system with Camera Link cameras has been developed to acquire and to reliably transport data at a throughput of 1000MB/sec. The image data are transferred through ZeroMQ protocol to the storage where the image data and the relevant metadata are archived and made available for user analysis. For high-speed frames of image data storage, optimized schema is identified by comparing and testing four schemas. The image data are written to HDF5 files and the metadata pertaining to the image are stored in NoSQL database. It could deliver up to 1.2GB/sec storage speed. The performances are also contrasted between a stand-alone server and the Lustre file system. And the Lustre could provide a better performance. Details of the image acquisition, transfer, and storage schemas will be described in the paper.

INTRODUCTION

Motivated by the successful operation of X-ray FEL facilities worldwide and the great breakthroughs in atomic, molecular, and optical physics, condensed matter physics, matter in extreme conditions, chemistry and biology, the first hard X-ray FEL light source in China, the so called Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), is under construction. SHINE will utilize a photocathode electron gun combined with the superconducting Linac to produce 8 GeV FEL quality electron beams with 1 MHz repetition rate [1].

A myriad of image data will be generated by the beam monitor system, the optical diagnostics system and the laser system, providing the required parameters for the accelerator operation and physics research. In order to measure the laser accurately, CCD cameras in the optical diagnostic system capture images at high speed. In addition, the beam cross section measured by the seed laser system and the drive laser system provides the basis for the commissioning and adjusting the devices ' parameters. Thus the accelerator has need of an efficient data acquisition and storage framework to accommodate the high-speed frames of image data, which is of great value to engineers and physicists to identify errors, component deterioration, poor process optimization, etc. They can also be used for big data analysis to improve control system stability and efficiency, and reduce maintenance cost. We have designed a general image system which is less expensive, using regular commercial hardware. It could acquire, transmit, and store the images at the speed of 1000MB/sec. The relevant tools are also developed to retrieve and display images on real time. Details are described in the following sections.

ARCHITECTURE

The whole system as shown in Fig. 1 can be divided into five functional modules, namely acquisition, transmission, storage, retrieval and online display.

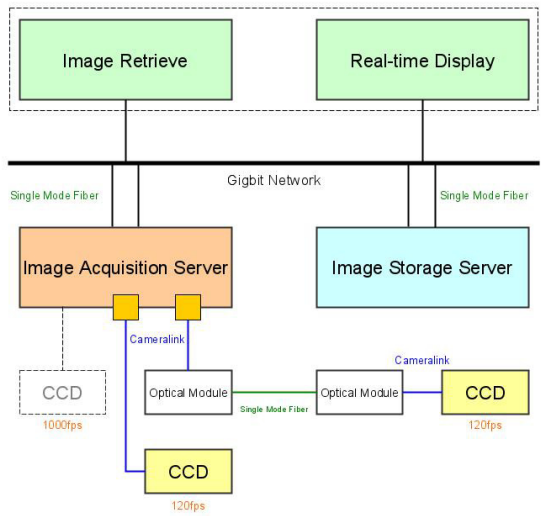


Figure 1: System Architecture.

Two CCD cameras with Camera Link interfaces take images at speed of 120 frames per second. One camera is directly connected to Camera Link Frame Grabber through a cable, while the other is connected to Camera Link Range Extender over a fiber cable, that solves distance limitation of Camera Link. Images are processed and packaged by the acquisition server and then transmitted to the storage server and online-display server through 10 Gigabit Ethernet, using ZeroMQ protocol for communication. After the storage server receives the data stream, it first unpacks it, then saves the image data in files as HDF5 format, and the metadata organized to facilitate searchability in MongoDB database. The web-based retrieval system is based on standard J2EE(Java 2 Platform, Enterprise Edition) Glassfish platform. It is designed to handle remote queries for historical records.

* Work supported by Shanghai Municipal Science and Technology Major Project(Grant No. 2017SHZDZX02)
† zhaohuan@zjlab.org.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

The online display server can display the images captured from two cameras in real time.

Acquisition Module

There are three commonly used camera interfaces as shown in Table 1.

Table 1: Camera Interfaces Comparison

Interface	Performance
GigE	slower speeds, 100m cable length
Camera Link	5.4Gbps, short cable length, cost effective
CoaXPress	6.25Gbps, short cable length, expensive

GigE is usually used in systems with less critical speed and timing demands. Camera Link offers higher link speed up to 5.6 Gb/s (Gigabits per second), but cable length is limited and cable cost is a bit high. CoaXPress supports the transfer rate up to 6.25Gb/s, and it is more expensive than Camera Link. Taking cost and performance into consideration, we choose Camera Link and utilize a fiber optic range extender to resolve distance limitation by connecting camera to frame grabber over single fiber cable.

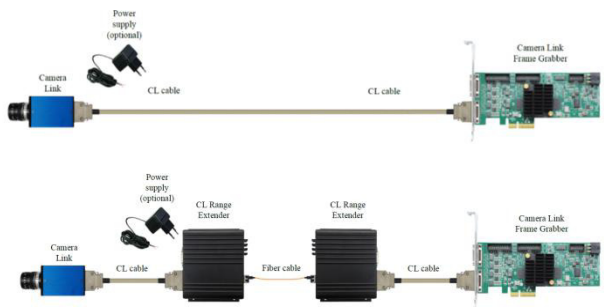


Figure 2: Camera Link Connection.

There are two cameras taking pictures at the same time. One is directly connected to Camera Link Frame Grabber through CL cable that is about 10 meters. The other is connected to CL Range Extender over single fiber cable that solves distance limitation of Camera Link. As shown in Fig. 2, the system is constructed of two converters, one on the camera side and one on the Camera Link frame grabber side. Camera Link Frame Grabber is a PCI Express-based device which allows high bandwidth communication between the device and the motherboard/server. The models of all devices are as shown in Table 2.

We use C language /Matrox Imaging Library(MIL) to develop programs for image capture, processing and annotation. MIL software development kit (SDK) is designed to reduce time and effort required to bring solutions. However, MIL SDK is mainly developed and utilized for Windows OS and has little support for Linux. It even cannot be installed in Linux. In order to install and use it in Linux, we tried multiple Linux versions, e.g., CentOS, Debian and Ubuntu. And finally we found Ubuntu 14.04 could match MATROX API.

Table 2: Devices List

Device Name	Performance	Number
CCD Camera	JAI SP-5000M-PMCL	2
Range Extender	Kaya Camralink Range Extender	2
Frame Grabber	Matrox_RAD EV 1G CLSF	2
CL cable		4
SFP module		2

Transmission Module

We make use of ZeroMQ[2] for image communication. ZeroMQ is a high-performance asynchronous messaging library, aimed at use in concurrent applications. ZeroMQ supports common messaging patterns (e.g., pub/sub, request/reply, client/server) over a variety of transports (e.g., TCP, in-process, inter-process, multicast and more), making inter-process messaging as simple as inter-thread messaging. We use pub/sub and request/reply patterns through TCP channels in the application.

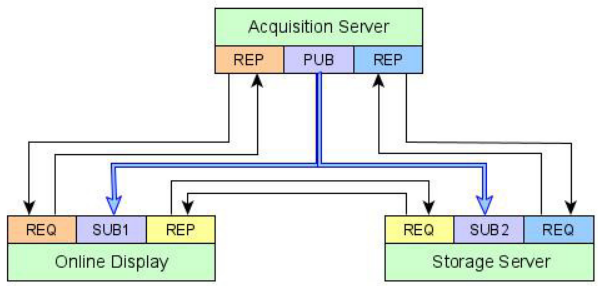


Figure 3: ZeroMQ Communication.

As shown in Fig. 3, request/reply connects the online display server (client) to the acquisition server (service), implementing the handshaking which should be successful before the transfer of data. Request/reply depends on a fixed send -> recv / recv ->send sequence. The connection will be interrupted if the sender does not receive the answer. Thus we modified two configuration parameters to realize that even if the sender does not receive a reply, it can continue to ask until the handshake is successful. The other two connections of acquisition server/storage server and display server/storage server are the same. Every two servers must shake hands to ensure the transmission synchronously.

Publish/subscribe connects the acquisition server and display/storage server. The publish-subscribe pattern is used for one-to-many distribution of data from a single publisher to multiple subscribers in a fan out fashion. Unlike request/reply pattern, messages are sent directly to the two servers, without the knowledge of what or if any subscriber of that knowledge exists. We use pub/sub pattern to transmit the stream of image data, allowing two clients to consume the stream.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

In order to get high transmitting speed,we modify the parameters of the ten gigabit network switch and servers, making multiple parameters match each other. After the modification, the transmission speed increases from 500MB/sec to 600MB/sec. To further increase the speed, we bind two network interfaces into a single logical 'bonded' interface as shown in Fig. 4. Network bonding increases the network throughout and bandwidth. Whereas network bonding increases CPU consumption and reduces overall performance. So we use two networks logically and in our program, data are transmitted through two networks at the same time. The total transmission speed increases to 1200MB/sec. That could satisfy our need.

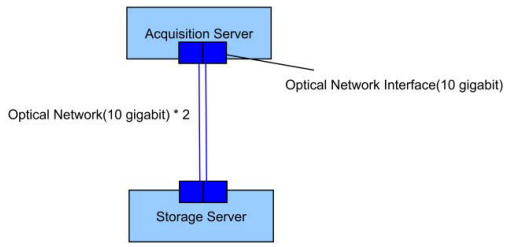


Figure 4: Network Bonding.

Storage Module

The image data represents an image, while the metadata pertaining to the image includes details relevant to the image itself as well as information about its production. What ' s more, the metadata is organized to facilitate searchability [3].

The image metadata involves:

- Number of the camera capturing the image
- Timestamp to the millisecond
- IP address
- Port number
- The camera resolution
- The depth of every pixels of the camera
- The gain of the camera
- The exposure time

We have designed four schemas for storage.

Storage Schema 1

The image data itself and the metadata are both stored in MongoDB. The image data is stored as a 2-D array of 8-bit unsigned integers.

Storage Schema 2

We make use of HDF5 as the permanent storage to store the image data, while using MongoDB as the index to store the metadata. The path where the image is stored is managed by MongoDB and used as the index to retrieve the HDF5 file.

Storage Schema 3

MongoDB is replaced by Cassandra.The metadata is stored in Cassandra as the index and the image data is stored as HDF5 format.

Storage Schema 4

Similar to Schema 2, we use DIRECT CHUNK WRITE strategy to write the data to HDF5 files. Others are same.

The storage performance is tested using three different sizes of gray level image, 1024×1024 bytes (1 MB),1024 × 2048 bytes (2 MB) and 2048 × 2048 bytes (4 MB), respectively. For each image, we record the time consumed to store 100 frames, 200 frames, 300 frames, 400 frames and 500 frames of images. The testing is performed in a rack-mounted server with 512 GB of RAM and 960 GB of SSD hard disk, running Linux/CentOS 7 operating system. There are 2 physical CPUs and each CPU has 6 cores in the server.

The test program is written to perform the same store operation 50 times constantly. Then we calculate the storage speed in MB/sec. The mean value with error bars is displayed in Fig. 5. The number of frames is on the x axis. The y axis corresponds to the storage speed in MB/sec. The image size is given on the bar. For instance, 1 M represents the image of 1024 × 1024 pixels (width: 1024, height: 1024) and each pixel is represented by a byte (8 bits).

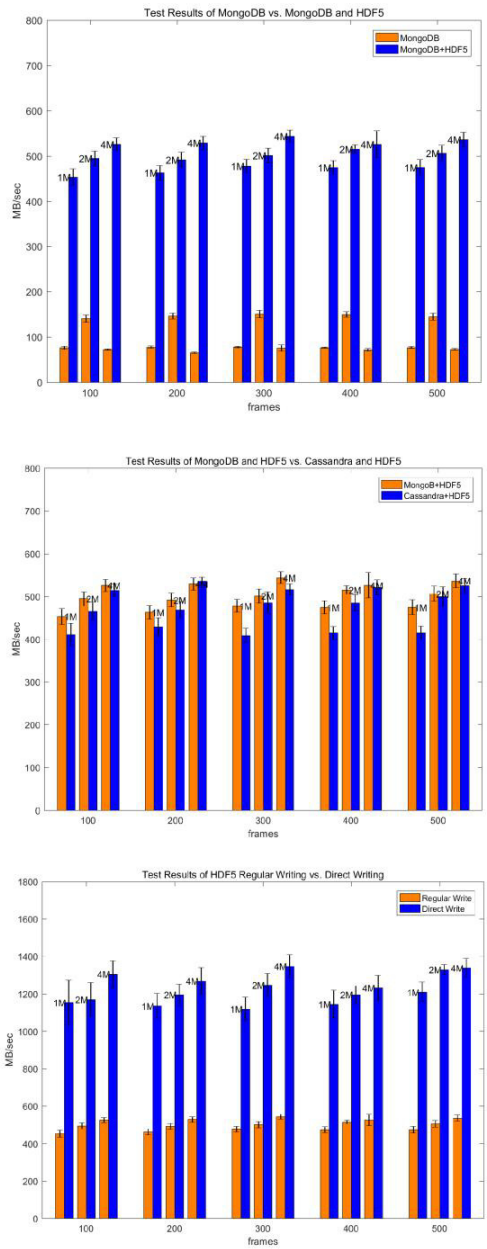


Figure 5:Test Results of Four Storage Schemas.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

As we can derive from Fig. 5, to use MongoDB to store the metadata and HDF5 DIRECT CHUNK WRITE to store image data provides the optimal storage performance, about 1200MB/sec. It seems to be difficult to increase the speed from the software design. So as to achieve a higher storage speed, we can consider to extend the hardware. However, the cluster provided by several servers could deliver a combined throughput. If we stream the image data to a cluster, it is likely to further improve the storage speed. Consequently, we use three servers to deploy a Lustre file system. There are a single metadata target (MDT) and two object storage targets (OST). MDT holds metadata information, which is stored separately from file object data. The data content is written to OSTs to persistent storage. Meanwhile, we configure two clients mounting to the Lustre file system using the 10 gigabit network protocol. The image storage performance is tested in the cluster. And we calculate the sum of the storage speed from two storage processes. It is about 1500MB/sec. The result shows that the cluster provides more optimal performance than the stand-alone server. While the Lustre in the test is a small scale distributed file system, which is composed of three servers mounted by two clients. The Lustre system can scale to large scale platforms provided by more servers, and present larger capacity storage space to more clients. Then we can obtain superior combined throughput performance. If there is higher requirement of data storage speed, the Lustre file system could scale easily in the future.

Retrieval Module

The retrieval system is based on Maven J2EE Glassfish platform with MongoDB and HDF5 utilized as backend data storage. As shown in Fig. 6, the application architecture diagram is composed by three parts: the persistence layer, the business logic layer and the client layer. The persistence layer is a general data storage container for both metadata and image data. The business logic refers to the processing demand to the data carried by the specific client. Data APIs are predefined database queries to facilitate application programming, which provides convenient access to the database. Data APIs use JDBC Connector to connect to MongoDB database to get a specific image's timestamp, cameroNo, index, etc. With index to HDF5 files, it uses Java HDF5 Interface (JHI5)[4] to access the image in HDF5 format. HTTPServlets process demand to the data carried by the web client and return data in JSON format. The client layer utilizes HTML5 Canvas to draw the pixel-level image, Bootstrap to style response websites and jQuery to handle the event handling and Ajax.

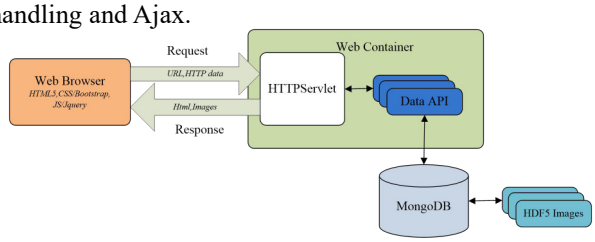


Figure 6: Retrieval Module Architecture.

Online Display Module

The online display module (Fig. 7) makes use of PyDM[5] to build graphic user interface. The image data transmitted through ZeroMQ protocol from the acquisition server are imported to the EPICS database located in the display server by PCASpy[6] interface. The total number of image frames acquired from the acquisition server is also displayed on this interface, compared with the number of images having been stored into the storage server (as shown in Fig. 8). The two are equal and it can indicate that there is no packet lost.

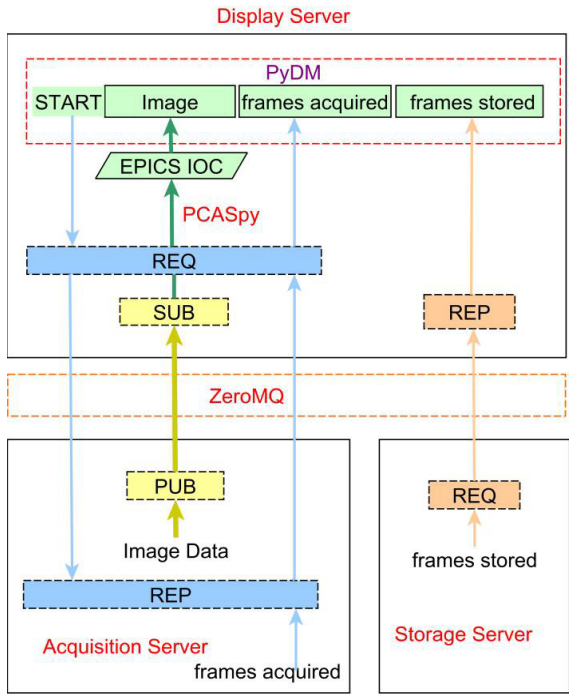


Figure 7: Online Display Module.

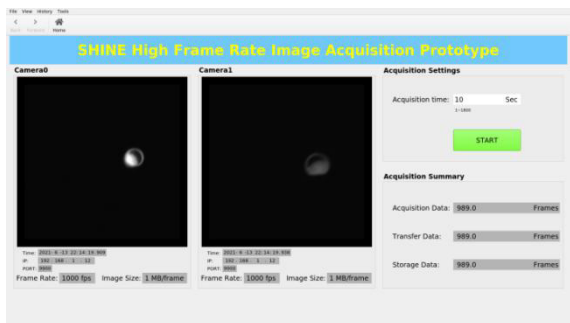


Figure 8: Online Display Interface.

TESTING

Test is taken by two cameras (120frames/sec, 1024*1024) illuminated by a diode which is excited by a sine wave signal generator as shown in Fig. 9. According to all images from one camera within two seconds, we find the brightest point of each image, the point with the largest gray value. Then we fit a sine curve using the least-squares method. The curve is shown in Fig. 10. The fitting frequency is equal to the frequency of the signal generator.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL02

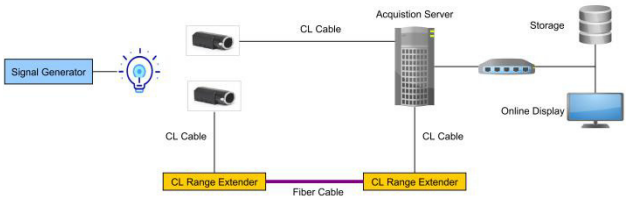


Figure 9: Testing.

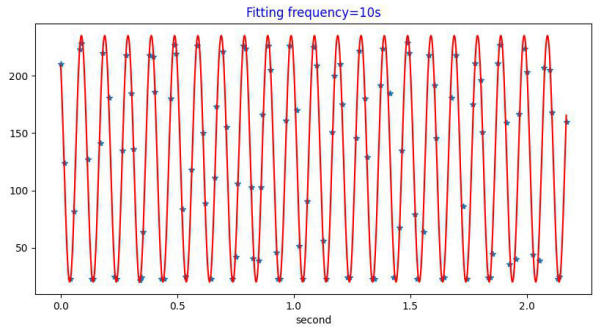


Figure 10: Fitting Curve.

We also test the performance of a higher speed, 1000 frames/sec with each image size 1MB. We monitor the number of HDF5 files generated every 5 seconds by Linux command ‘watch -n 5 "ls |wc -l |tee -a num.log"’. The result is shown in Fig. 11. Then the storage speed could be calculated, 500MB * 10 / 5sec = 1000 MB/sec. It shows that our system can get the speed of 1000MB/sec.

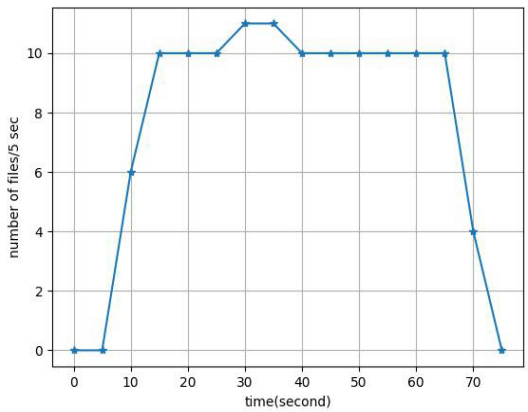


Figure 11: Test Results of 1000 Frames/sec.

CONCLUSION

The system is able to acquire, transmit and store the image data at speed of 1000MB/sec stably without loss. The solution is based on Camera Link interface to capture high-frame rate images, ZeroMQ protocol to transmit stream data, together with HDF5 to store the heavy image data and MongoDB for indexing the heavy data in HDF5. Multi-threading and network binding increases the holistic system performance as well. The hardware architecture and software design is not limited to image data. It could also manipulate the waveform data for SHINE.

REFERENCES

[1] Kai Li, Haixiao Deng, “Systematic design and three-dimensional simulation of X-ray FEL oscillator for Shanghai coherent light facility”, *Nucl. Instrum. Methods,* vol. 895, pp. 40–47, 2018. doi:10.1016/j.nima.2018.03.072

[2] https://zeromq.org/

[3] Lv H., Yan Y., Wang H., “The data storage system for SHINE”, *Nucl. Instrum. Methods,* vol. 1002, 2021. doi: 10.1016/J.NIMA.2021.165285

[4] https://www.hdfgroup.org/

[5] http://slaclab.github.io/pydm/

[6] https://pypi.org/project/pcaspy/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

MANAGE THE PHYSICS SETTINGS ON THE MODERN ACCELERATOR*

T. Zhang†, K. Fukushima, T. Maruta, P. Ostroumov, A. Plastun, Q. Zhao
Facility for Rare Isotope Beams, Michigan State University, East Lansing, USA

Abstract

The Facility for Rare Isotope Beams (FRIB) at Michigan State University is a unique modern user facility composed of a large-scale superconducting linac capable of accelerating heavy-ion beams from oxygen to uranium. An advanced EPICS-based control system is being used to operate this complex facility. High-level physics applications (HLAs) developed before and during the staged commissioning of the linac are one set of the critical tools that resulted in achieving the commissioning goals quickly, within several shifts. Many of these HLAs are expandable to other EPCIS controlled accelerators. Recently developed HLAs deal with the management of extensive data to achieve the repetitive high performance of ion beams in the entire linac measured by non-destructive diagnostics instruments, and open the possibilities to explore the extra values out of the data. This paper presents our recent significant development and utilization of these HLAs.

INTRODUCTION

The LINAC of the Facility for Rare Isotope Beams project at Michigan State University is a unique modern superconducting accelerator, it is designed to deliver all the stable isotope beams with multiple charge states to the kinetic energy higher than 200 MeV per nucleon, and power of 400 kW on the target [1]. Since the mid of 2017, staged commissioning has achieved a series of remarkable successes [2–4]. Now FRIB project is approaching the next milestone, that is to generate and separate the rare isotope beams at and after the production target system.

To expedite the beam commissioning, various kinds of high-level physics applications (HLAs) have been developed and deployed to FRIB controls network. The Python-based software framework called `phantasy` is designed and developed to drive the entire high-level communication between the accelerator and physicists [5]. Based on `phantasy`, the Python interactive scripting environment is implemented and utilized to prototype the physics tuning algorithms, and also used to control the machine in the advanced expert mode. Graphical user interface (GUI) applications have been developed with well-tested physics algorithms to make machine tuning simple and confident. General tools and GUI widgets based on Qt framework [6], together with physics-related widgets have been developed as another major part of `phantasy` framework to streamline the GUI application development.

* Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University.
† zhangt@frib.msu.edu

To better organize the data generated in the controls network, software applications have been developed to ensure the quality of data, e.g. integrity, reliability, and availability, etc. The next coming sections will present the software development activity of FRIB high-level physics controls, the use cases of the development on the accelerator of FRIB, and the approach of how the physics data is managed.

THE EVOLUTION OF `PHANTASY`

Originally, `phantasy` was designed for the object orient high-level controls for EPICS-based accelerator system. The main goal was to abstract the entire machine to be controllable in the ecosystem of Python, by using the enormous third-party library to fulfill the machine tuning missions [7].

Several critical issues need to be addressed with neat solutions before accomplishing such high-level conrtols environment:

- Properly present the machine in the view of computing environment.
- Properly handle the value of physics quality in different scenarios, either from the view of device control or physics model.
- Properly interface with the physics model, which is usually separately developed to simulate the accelerator behavior.
- Properly manage the development resources, i.e. source code, support data files, testing, deployment, etc.

With `phantasy`, the physics model-independent machine representation could be generated in the view of object-oriented. The machine is represented as a series of devices, each device is an instance of a general abstracted high-level element class, which is composed of controllable and non-controllable attributes. The non-controllable attributes usually present the static properties of the device and the controllable ones are connected to the process variables which are served through EPICS IOCs. The device control is fulfilled via Python object getter and setter operations, through dotted-syntax. All the device information is maintained separately, such a way the core code of `phantasy` could also be working with other accelerator systems, the same strategy has been applied to the GUI application development.

The interpretation of the values for physics qualities is handled in another separated application called UNICORN [8], which is a web application that features REST API. The Python interface "python-unicorn" [9] is also developed to request either value in physics or engineering unit. The mapping rules between physics and engineering world could be defined based on the needs of the physics model, e.g. for

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the dipole, the current in Ampere could be interpreted to Tesla as the magnetic field strength, or Tesla-Meter as the integral strength, thus two rules could be defined in UNICORN, use `.B` to read the device setting in Tesla, and `.B = <new-value>` to set with new magnetic field in Tesla, it is also possible to read it into Tesla-Meter via `.TM` if it is defined in the device configurations.

Speaking to the model interface, `phantasy` now supports modeling the accelerator with `FLAME` [10]. The internal virtual accelerator component is also developed with such a linear fast executing model. On top of the abstracted device control layer, any physics tuning algorithms could be prototyped quickly in any Python terminal environment, Jupyter [11] is used as the main one. `phantasy` is also designed with the ability to adapt with other physics models, to do that, the routine of model-dependent machine representation needs to be developed.

All the software development is managed with modern tools. `GIT` [12] is used to manage the source code, the whole physics application development is managed with the internal Bitbucket service [13]. More than 20 Debian packages are built to deploy the development into FRIB controls network via the automated CI/CD infrastructure [14]. Accumulated code changes will be released as new versions of software and deployed into production through the automated pipeline.

Recently, `phantasy` has been enhanced a lot on the GUI application development side. New GUI widgets have been developed for PyQt application development. To unify the UI style and make use of existing resources, command-line tools also have been developed and improved. One can initiate the app skeleton by "makeBasePyQtApp" command. It is the plan to release all the development through PyPI [15] (some of the packages have already been published there) and PPA [16] in the near future, along with the documentation sites, which require a lot of effort.

MANAGE THE DATA IN THE CONTROLS NETWORK

From the view of the high-level physics controls, there are different kinds of data distributed and generated in the controls network.

The first category is device information, e.g. the name, size, location, controls process variables, etc., all these information is already managed in `phantasy` framework, with above-mentioned dotted-syntax, all these attributes could be reached naturally in Python environment, as well as IOC application.

The second category is the data from other web services. For instance, the data from directory service ChannelFinder [17], the data from alarm services, the data from Archiver Appliance [18], the data from Olog [19], and the data from UNICORN service, etc.

The third category is the data from various software IOCs. When building the high-level physics controls system, software IOC applications are developed to fulfill the specific data processing tasks, the results are either integrated into `phantasy` or exposed as PVs.

The fourth category is the data from physics models, which could be lattice files, simulation results.

Last but not least one is the developed software itself.

On the other side, in the view of machine tuning, all these data should be well-organized to achieve the following goals:

- The developed machine tune should be accurately saved.
- The saved machine tune should be accurately restored.
- There must be some way to verify the restored machine tune.
- There must be some way to easily apply one machine tune to different charge states and ion species.

To address them, systematic software development is undergoing.

Physics Settings

Here the data of physics settings is defined as a series of device set values along the accelerator beamline. The app named "Settings Manager" has been developed to manage the physics settings. "Settings Manager" is one of the GUI apps that is built upon `phantasy` framework, it can work with different machine/segment, which is defined in "phantasy-machines" (a package that keeps all the configurations of machine/segment for `phantasy`).

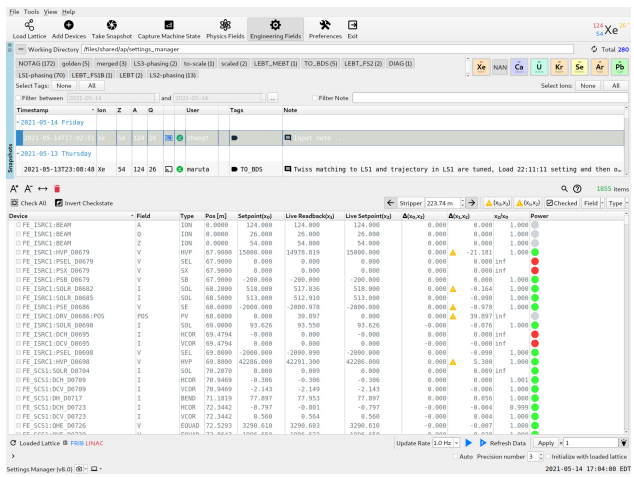


Figure 1: Main user interface of "Settings Manager".

Figure 1 shows the main user interface of "Settings Manager". Buttons in the toolbar indicate the features it provides, as well as the current working ion species shown at the rightmost. Right below the toolbar is the area for saved physics settings or snapshots. All the snapshots are sorted by the saved time by default, all come with the ion species info, saved user name, tags, and note. "Tags" is a string of words separated by a comma, to label the snapshot data, and "Note" is a string for arbitrary additional info about this snapshot.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

The expanded snapshot area is showing in Figure 2, it features tabular view of all the snapshots, supports sorting and filtering, and convenient buttons for filtering by ion names or tags, to quickly locate the snapshot entries.

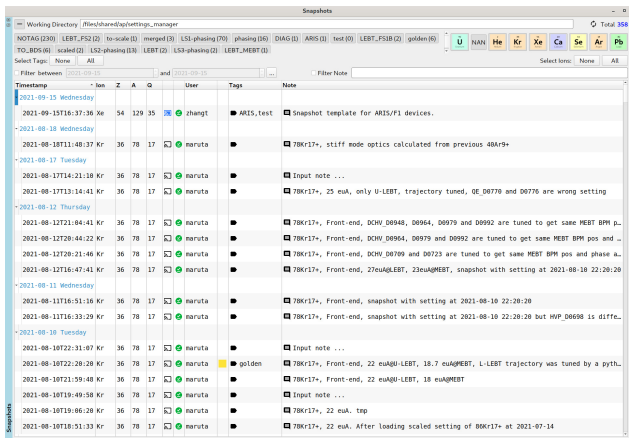


Figure 2: Snapshot management window of "Settings Manager".

The snapshots could be loaded and presented in the right bottom area by double-clicking or right-clicking context menu actions. In Figure 1, one can see the listed device settings, each row is for one "Field", each "Field" is a control knob, the device info could be reached via the context menu on the "Device" column. The saved device settings are shown in column "Setpoint(x_0)", while current settings are in column "Live Setpoint(x_2)", column "$\Delta(x_0, x_2)$" indicates how they are matched or not.

The procedure for applying saved settings to machine is, first check the fields that need to be applied (check by clicking, keyboard and context menu actions), then press "Apply" button to do process one by one, the details of device settings could be controlled via the "Preferences" menu. One feature to support multiple charge states settings is applying with a scaling factor, which is right after "Apply" button, the value will be automatically calculated, but also could be modified, default is 1.0. After applying all the settings, recheck column "$\Delta(x_0, x_2)$" to make sure the current settings are updated accurately, for the case of applying with a scaling factor, check column "x_2/x_0", which is expected to be the value of the scaling factor.

The data model of the snapshot supports exporting to different kinds of data files for portability, typical data formats are XLSX and HDF5, from which one can have the metadata and physics settings data, as well as the machine state data.

Machine state data is to address how to verify the machine performance under one specific physics settings. The definition of it is the readings from various diagnostics devices when taking the snapshot. The configuration file for machine state data capturing is a predefined file shown in Figure 3, could work with CLI tool `fetch_mach_state` (which is deployed with `phantasy-apps` package) as well. When taking a snapshot, the machine state data is first captured based on the configuration file, then capture current

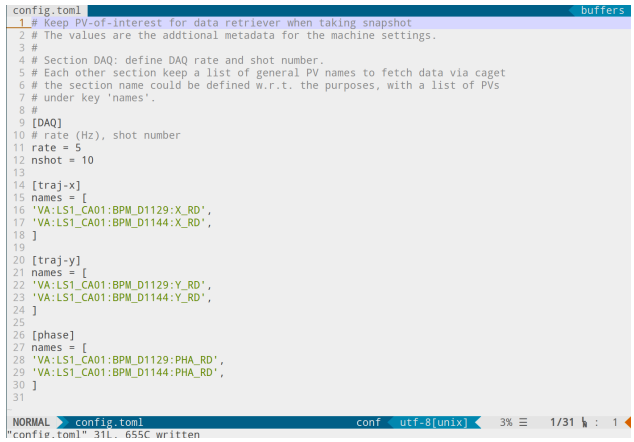


Figure 3: An example of the configuration file for machine state fetching.

physics settings, and save them together into one snapshot, finally shown as one entry in the snapshot window. "Settings Manager" supports managing snapshots in a database or a directory. The machine state data also could be captured at any time via the tool in the toolbar. The obvious benefit of saving physics settings with machine state is that all the data actually been accurately "labeled", which is very valuable for the future machine learning application.

Figure 4 visualizes the trajectory along the FRIB LINAC from the machine state data, one can also visualize other qualities, e.g. BPM phase, amplitude, to verify the physics settings.

It is also worth mentioning that the physics settings are also pre-generated from physics models before the beam commissioning and imported into "Settings Manager" for use, which significantly speeds the beam tuning efficiency.

ARCHIVED DATA

Another essential data source for the physics application software is the data archiver. FRIB is using the site-maintained Archiver Appliance application [18] to keep archived PV values. The archiving and data retention policies are defined in each IOC. The time-series data served by Archive Appliance could be visualized in CS-Studio data browser application.

To integrate the archived data into the Python ecosystem, a dedicated Python client package has been developed, called "pyarchappl" [20], which is not only designed for communicating with Archiver Appliance via REST API, but features with data-wrangling capabilities, to work with the archived data with popular data science tools.

One of the useful command-line tools distributed with `pyarchappl` is `pyarchappl-get`, which is a general command for data retrieval from Archiver Appliance. Figure 5 shows the user guide of the command. One can easily get timestamp aligned archived PV data in the form of CSV, XLSX and HDF5.

The retrieved data set could be processed by reading the data file. Figure 6 is the heat map view for the x and y

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

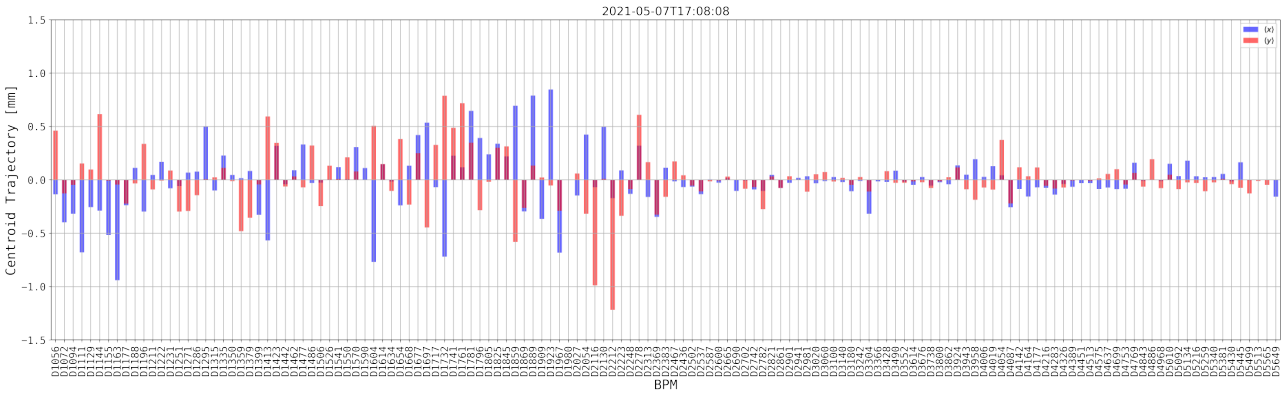


Figure 4: Typical example of trajectory visualization with BPM data

```
tong ~ > pyarchappl-get -h
usage: pyarchappl-get [-h] [--url URL] [--pv PV_LIST] [--pv-file PV_FILE]
                      [--from FROM_TIME] [--to TO_TIME] [--resample RESAMPLE]
                      [--verbose] [-o OUTPUT] [-f FMT]
                      [--format-args FMT_ARGS]

Retrieve data from Archiver Appliance and export as a file.

optional arguments:
  -h, --help            show this help message and exit
  --url URL             URL of Archiver Appliance, default is FRIB FTC
                        archiver (default: None)
  --pv PV_LIST          List of PVs for retrieval, each define with --pv
                        (default: None)
  --pv-file PV_FILE     A file for PVs, one PV per line (skip line starts with
                        #), append each to pv_list (default: None)
  --from FROM_TIME      A string of begin time in ISO8601 format (default:
                        None)
  --to TO_TIME          A string of end time in ISO8601 format (default: None)
  --resample RESAMPLE   The offset string/object representing target
                        conversion, e.g. '1S' for resample with 1 second
                        (default: None)
  --verbose, -v         Verbosity level of the log output, 0: no output,
                        1(-v): output progress, 2(-vv): output progress with
                        description (default: 0)
  -o OUTPUT, --output OUTPUT
                        File path for output data, print to stdout if not
                        defined (default: None)
  -f FMT, --output-format FMT
                        File format for output data, supported: csv, hdf,
                        excel, html, ... (default: csv)
  --format-args FMT_ARGS
                        Additional arguments passed to data export function in
                        the form of dict, e.g. '{"key":"data"}' (for hdf
                        format) (default: {})

Examples:
# Retrieve raw PV data in the defined time frame
$ pyarchappl-get -o data.csv -v \
  --pv LS1_CA01:BPM_D1129:XPOS_RD --pv LS1_CA01:BPM_D1129:YPOS_RD  \
  --from 2021-04-15T20:10:00.000Z --to 2021-04-15T21:25:00.000Z \

# Align the timestamps, resample at 1 second
$ pyarchappl-get -o data.csv -v \
  --pv LS1_CA01:BPM_D1129:XPOS_RD --pv LS1_CA01:BPM_D1129:YPOS_RD  \
  --from 2021-04-15T20:10:00.000Z --to 2021-04-15T21:25:00.000Z \
  --resample 1S
```

Figure 5: Help message of `pyarchappl-get`.

readings of the 75 BPMs along the accelerator through first LINAC segment (LS1) to the first fold segment (FS1), in the time range from 16:00 to 20:00 EDT on Oct. 13, 2020, total 72000 data entries. The blue boxed areas are the minimal absolute trajectory readings, which is within the range of ±0.5 mm after trajectory correction had been performed.

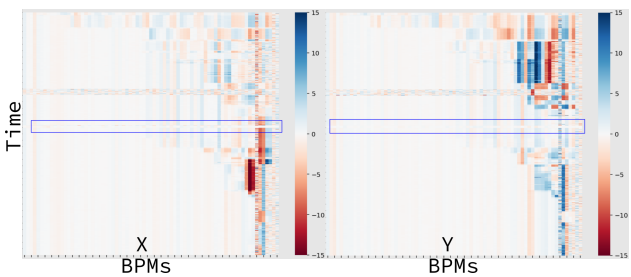


Figure 6: Heat map plot of time-series archived data for central trajectory in x and y.

Tools also have been developed to retrieve the physics settings at the time when the trajectory reached the minimal, then the retrieved data is processed to be a new snapshot for "Settings Manager". All these tools will be integrated into "Settings Manager" as new context menu actions.

CONCLUSION

The high-level physics controls software framework `phantasy` is continuously moving forward. The enhanced toolkits for PyQt GUI development greatly streamlined the HLA development. The solid software solution has been designed and implemented into the data management of FRIB LINAC commissioning. "Settings Manager" has been developed and used to accurately manage the physics tunes for the entire LINAC, as well as keep the machine state dataset to make the data of physics settings more valuable for other applications. "Settings Manager" expedites the LINAC tuning for multi-charge state operating mode. The Python client package "pyarchappl" for Archiver Appliance has been designed and developed for easy data retrieval and general data analysis with Python third-party packages. The data management solution is continuously consolidating to prepare for the concrete machine learning applications.

ACKNOWLEDGMENTS

The authors would like to thank A. Carriveau and T. Ashwarya for the useful discussions.

REFERENCES

[1] J. Wei *et al.*, "Advances of the FRIB project", Int. J. Mod. Phys. E 28, 1930003 (2019). doi:10.1142/S0218301319300030

[2] P. Ostroumov *et al.*, "Heavy ion beam acceleration in the first three cryomodules at the Facility for Rare Isotope Beams at Michigan State University", Phys. Rev. Accel. Beams. 22, 040101 (2019). doi:10.1103/PhysRevAccelBeams.22.040101

[3] P. Ostroumov *et al.*, "Beam commissioning in the first superconducting segment of the Facility for Rare Isotope Beams", Phys. Rev. Accel. Beams. 22, 080101 (2019). doi:10.1103/PhysRevAccelBeams.22.080101

[4] P. Ostroumov *et al.*, "First Simultaneous Acceleration of Multiple Charge States of Heavy Ion Beams in a Large-Scale Superconducting Linear Accelerator", Phys. Rev. Lett. 126,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

114801 (2021).
doi:10.1103//PhysRevLett.126.114801

[5] T. Zhang *et al.*, "High-level Physics Controls Applications Development for FRIB", in *Proc. 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS 2019)*, New York, NY, USA, Oct. 2019, pp. 828–834. doi:10.18429/JACoW-ICALEPCS2019-TUCPR07

[6] Qt GUI framework, https://www.qt.io

[7] T. Zhang, "Physics high-level applications and toolkit for accelerator system", in *EPICS Collaboration Meeting*, Argonne National Laboratory, IL, USA, Jun. 2018. https://epics.anl.gov/meetings/2018-06/talks.html

[8] Web application for physics quantity interpretation, https://github.com/phantasy-project/unicorn-webapp

[9] Python interface to UNICORN, https://github.com/phantasy-project/unicorn

[10] Z. He *et al.*, "The fast linear accelerator modeling engine for FRIB online model service", Computer Physics Communications 234, 167 - 168 (2019). doi:10.1016/j.cpc.2018.07.013

[11] Jupyter, https://jupyter.org

[12] Git, https://git-scm.com

[13] Bitbucket, https://bitbucket.org

[14] M. Konrad *et al.*, "Automatic deployment in a control system environment", in *Proc. 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS 2019)*, New York, NY, USA, Oct. 2019, pp. 126–131. doi:10.18429/JACoW-ICALEPCS2019-MOMPL006

[15] The Python Package Index, https://pypi.org

[16] Personal Package Archive, https://launchpad.net/ubuntu/+ppas

[17] ChannelFinder, http://channelfinder.github.io

[18] Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs

[19] Online logbook service, https://github.com/Olog

[20] Python client to Archiver Appliance, https://github.com/archman/pyarchappl

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

FAIR MEETS EMIL: PRINCIPLES IN PRACTICE

Gerrit Günther*[1], William Smith[1,2], Markus Kubin[1], Marcus Bär[1–4], Nico Greve[1], Rolf Krahl[1], Simone Vadilonga[1], Regan G. Wilks[1,2], Oonagh Mannix[1],
[1] Helmholtz-Zentrum Berlin für Materialien und Energie GmbH (HZB), Lise-Meitner-Campus, Hahn-Meitner-Platz 1, 14109 Berlin, Germany
[2] Energy Materials In-Situ Laboratory Berlin (EMIL), HZB, Albert-Einstein-Str. 15, 12489 Berlin, Germany
[3] Helmholtz Institute Erlangen-Nürnberg for Renewable Energy (HI ERN), Albert-Einstein-Str. 15, 12489 Berlin, Germany
[4] Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), Egerlandstr. 3, 91058 Erlangen, Germany

Abstract

Findability, accessibility, interoperability, and reusability (FAIR) form a set of principles required to ready information for computational exploitation. [1] The Energy Materials In-Situ Laboratory Berlin (EMIL) at BESSY II [2], with its unique analytical instrumentation in direct combination with an industrially-relevant deposition tool, is in the final phase of commissioning. It provides an ideal testbed to ensure workflows are developed around the FAIR principles; enhancing usability for both human and machine agents. FAIR indicators [3] are applied to assess compliance with the principles on an experimental workflow realized using Bluesky. [4] Additional metadata collection by integrating an instrument PID [5], an electronic laboratory book, and a sample tracking system is considered along with staff training. Data are collected in Nexus format and made available in the ICAT repository. This paper reports on experiences, problems overcome, and areas still in need of improvement in future perspectives.

INTRODUCTION

The FAIR principles – findability, accessibility, interoperability and reusability – are well known and serve as guidelines for data curators and data management stewards along with IT professional; people who manage rather than generate data. Here, we apply high-level data management concepts contained in the FAIR principles on the level of granularity of a complex, beamline-based materials research infrastructure – the EMIL infrastructure at BESSY II, that is in its final phase of commissioning. In doing so we clarify the relationship between data generator, research infrastructure, and data infrastructure; demonstrating how the FAIR principles relate to each actor.

To report the present state of work, the remainder of this paper is organized as follows: First, we sketch the complex scientific infrastructure of EMIL at BESSY II focusing on its end station SISSY I (**S**olar energy material **I**n-**S**itu spectroscopy at the **Sy**nchrotron). Since the FAIR principles are independent of each other [1] we continue with a discussion of their implementation following the sequence reusability, interoperability, accessibility, and findability. By balancing user effort and data accuracy we proceed to focus on the sequence of measures taken to find an appropriate workflow. Finally, we apply FAIR indicators and assess the SISSY I end station.

Throughout the paper, the convention of [3] is adopted to distinguish between data and metadata. Here, data refers to the primary output of detectors or other objects of outstanding scientific interest while metadata belongs to information that helps to analyze the primary data such as the sample or the sample's properties. If both types of data are addressed the term (meta)data is used.

OUTLINE OF EMIL INFRASTRUCTURE

The Energy Material In-Situ Laboratory Berlin (EMIL) [2,6,7] is a newly implemented, complex research infrastructure at BESSY II which combines state-of-the-art synthesis and deposition systems on industry-relevant scale with sophisticated in system/in situ/operando x-ray spectroscopy capabilities, outlined in Fig. 1. By focusing on study of the growth of materials and the formation of functional interfaces EMIL tracks the evolution of material properties through successive processing steps. A fully automated ultra-high vacuum multi-chamber system built by PREVAC [8] transfers samples between various production sites, analysis tools, and the SISSY I end station. As a result samples are changing both their state and location within EMIL and may be placed at in SISSY I end station for characterization.

The SISSY I end station houses an electron analyzer able to detect electrons with kinetic energies between 50 eV and 10 keV. Exploiting the full energy range of EMIL's two-color beamline, photon energies between 80 and 10000 eV – delivered by the two canted undulators (UE48 and U17) – can be used to probe the sample properties by soft and hard x-ray photoelectron emission electron spectroscopy. [9] Currently, the SISSY I end station is in the final stages of commissioning and so provides an excellent test bed to implement the FAIR principles during the transition to user operation.

The instrument and beamline are controlled with EPICS [10]. Devices are accessed in a Python environment through

* gerrit.guenther@helmholtz-berlin.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

the abstraction layer Ophyd [11]. The Python library Bluesky [4] uses this abstraction layer to operate the instrument and beamline hardware. Measurement data and metadata are saved together in a mongoDB database [12], which is not publicly accessible. A more detailed outline of the architecture is found in [13].

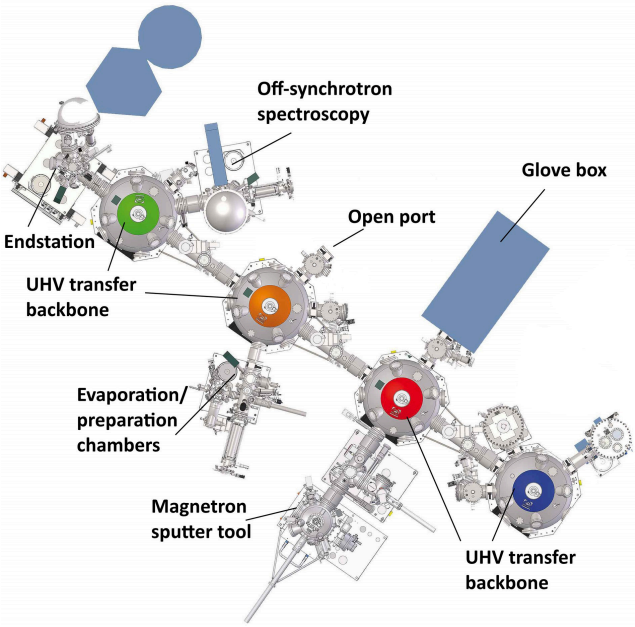


Figure 1: Schematic presentation of the SISSY@EMIL's setup connecting various processing and analysis chambers, among them the SISSY I endstation.

REUSABILITY – RICHNESS OF (META)DATA

The SISSY I operation is envisioned to be highly automated. It provides well-defined experimental environments enabling the automatic collection of detailed (meta)data. Unfortunately, not all critical metadata can be collected automatically and crucial information such as sample name, calibration measurements, special setups, or notes during the experiment may be buried in a paper log book and run the potential risk of being permanently lost.

Sample Information

In most cases sample information cannot be automatically collected but is critical for data reuse. For that reason, we implement the collection of sample information on the instrument level by defining a sample pseudo-axis in the Bluesky control software of SISSY I to enter the sample name, chemical formula, InChi key [14], or another sample identifier. The sample axis is treated in the same way as any motor position which would allow to define the sample by a script running for a lengthy period for example, when a sample holder with several samples is used.

At SISSY I all samples are transferred through the EMIL vacuum transport system, a highly automated environment in which samples may undergo successive steps of physical treatments and analysis. Each operation is stored in a SQL database and therefore the entire history of a sample is accessible. We implemented a readout routine to update the sample-pseudo axis on demand. If necessary, this can be corrected or enriched by the user during the experiment. It automatically identifies the sample holder currently at SISSY I through the sample database by using the date and time of the measurement. Once the sample holder is identified, the sample information is extracted from the sample database and former entries of the sample are collected in a chronological list of applied processes and measurements. In case the initial state of the sample when entering EMIL corresponds to an industrial standard (such as a silicon wafer) the sample would be fully defined.

For other cases we provide workarounds to add information manually. We implemented a graphical user interface to help describing the sample and enrich (meta)data by additional files such as photos, notes, or documents after the measurement. Further, we implemented a routine to trawl pdf documents from the HZB's proposal system GATE for samples which can be selected from a drop-down menu to add their information. However, direct access to the proposal system would be advantageous and has to be established.

Electronic Logbook – Another Kind of Metadata

Paper logbooks are still used at most BESSY II beamlines and contain useful or, even essential information for scientific reproducibility. Details of sample preparation, other analytical results, or even the experimental methodology can be found in these log books; additional qualitative information such as the scientists thoughts at the time of measurement, or classification in a wider context can also be found. Such intellectual interpretation of data belongs to another type of metadata that cannot be harvested automatically – but that can be stored electronically. To provide an electronic option of paper notebooks an instance of the ESRF e-logbook [15] was installed as a BESSY II-wide service. The logbook can include comments from the experimental team, as well as comments written automatically by the machine. Information about the current plan being executed is written by Bluesky in a callback.

Since Jupyter notebooks enjoy great popularity in the community to combine code, notes, and visualizations of results within a single file [16], and they are native to the Python Bluesky environment, their integration in a reproducible way is under discussion.

Persistent Identifiers

The provided information should be unambiguous; using globally unique persistent identifiers ensures clarity when considering important metadata. For example, an ORCID [17] distinguishes multiple individuals clearly even if they share the same name. For this reason, at SISSY I ORCIDs are automatically assigned to members of the experimental team if they are identified through name and facility.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Moreover, globally unique instrument PID [5] identify the instrument SISSY I [18], the soft [19] and hard [20] x-ray EMIL beamlines, and other parts of EMIL [21,22] to clearly state from where the data originates and where the instrument belongs. In the future, more PIDs will be included to precisely categorize important metadata, not least a semi-automated, globally unique sample identification system.

INTEROPERABILITY – SPEAKING THE SAME LANGUAGE

SISSY I stores metadata parameters acquired by Bluesky in a Mongo database. The names of these parameters are specific to this facility, beamline, and endstation. So that this metadata is eventually understandable to a wider audience we implemented a routine that converts the database content of a measurement to the NeXus file format [23] which is a well-known standard of the x-ray and neutron community. NeXus uses HDF5 [24] as physical file format and applies the NeXus Definition Language (NXDL) to clearly define the nomenclature and arrangement of information while being versatile enough to meet the demands of a wide range of complex scientific techniques.

The file format is suitable for high-performance data processing [25, 26] and is supported by capable analysis software such as Dawn [27] which provides a Python interface following the spirit of the Bluesky control software.

ACCESSIBILITY – TWENTY-FOUR/SEVEN

Well-curated data is useful when revisiting work after some time or when continuing another's work, but unleashes its full potential when accessible to co-workers, instrument users, and collaborators from around the world. That is why NeXus files are stored in HZB's central data repository [28] which is an instance of the ICAT software [29, 30] also in use at other Photon and Neutron facilities, including ISIS, Diamond Light Source, and ESRF.

ICAT's authentication system allows the experimental team to access the files associated with its proposal through a web interface using the HTTP standard protocol. According to the HZB data policy [31] raw data and associated metadata become openly accessible after a embargo period of normally five years.

However, the HZB data repository stores data on a tape library which has drawbacks concerning accessibility in general. In order to access the data, a request must be made in the metadata catalogue. The data will then be staged to disk automatically and a download link will be provided once the staging is complete. This might cause accessibility barriers to machine-agents, but there is currently no rigid procedure to deal with the situation.

FINDABILITY – 'HELLO, WORLD'

SISSY I's local (meta)data storage procedure using Mongo database is more advanced than most current BESSY II instruments that store data and metadata in various files. The Mongo database is completely searchable and returns measurements and metadata without laborious scans of a (potentially) confusing file structure. The same advantage applies to the ICAT's metadata catalogue which stores selected (meta)data of ingested files in a separate database and is also searchable. The difference is that ICAT is publicly accessible and centrally managed.

To increase visibility we need higher level services than those of HZB to connect (meta)data and the world – we need globally unique persistent identifiers (PID) to give measured raw data an unambiguous identity. While for (processed) data publications a manual procedure for adding PIDs exists, ingested raw data files in ICAT thus far lack this feature. However, the automatic assignment of PID's to single NeXus files containing raw data is a prerequisite to be harvested by higher level catalog services such as B2Find [32] and is currently in the implementation phase. This and the Harmonization of the metadata schema with regard to B2FIND is work in progress and ensure that (meta)data collected at SISSY I will be globally findable once the embargo period has ended. Such high-level search engines are crucial hubs in the world-wide data network and make the instrument's work public to visible to an community beyond that of the repository.

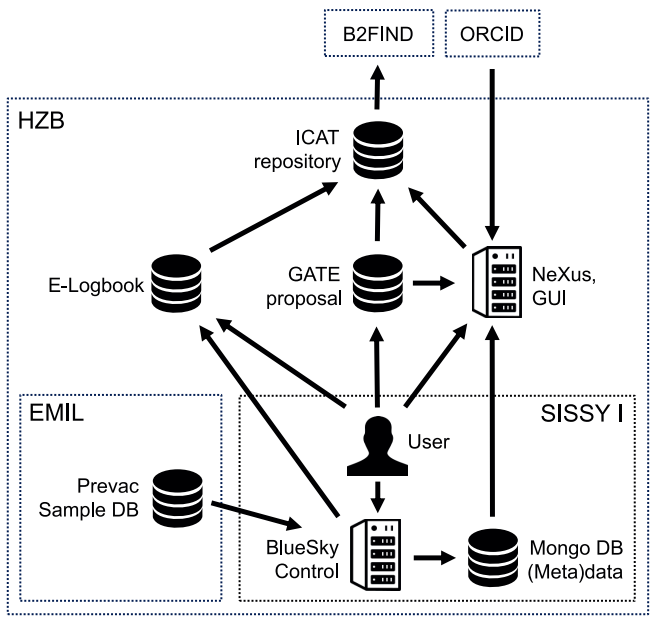


Figure 2: Visualization of the data infrastructure connecting the SISSY I instrument with services inside and outside the HZB.

WORKFLOW

In the previous sections, we presented tools and services that are implemented or under consideration. We purposefully did not define a unique sequence of executed steps. Since different experiments involve different procedures, we adapt the workflow to the needs of each experiment.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

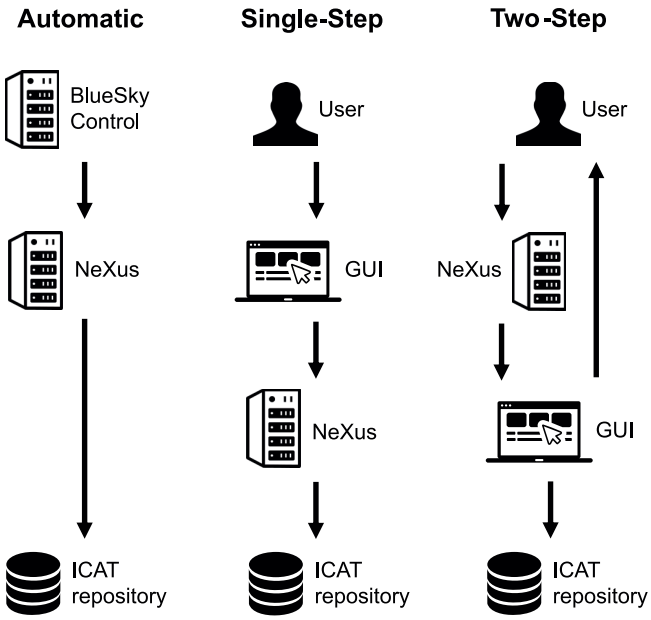


Figure 3: Provided workflows for the SISSY I instrument including automatic and manual processes.

Metadata may become available at different times of the experiment or even after its completion and thus the data management process may become stuck somewhere requiring human input to make data meaningful.

However, general principles guide the implementation of each workflow. Whenever possible procedures are outsourced to a physically separated server to keep the impact of the data management process on the experiment to a minimum. The parallel, mostly independent infrastructure is suitable to update running instruments without interrupting the production of results which is a major problem in fragmented software landscapes such as BESSY II [33].

The central data conversion server runs (i) the readout routine to get (meta)data from SISSY I's Mongo database, (ii) optionally the readout routine to collect sample information from the sample database, (iii) the routine trying to assign ORCIDs to members of the experimental team, and (iv) the NeXus writer routine that creates a NeXus file and feeds the file to the ICAT repository (see Fig. 2). All routines are written in Python, to harmonize with SISSY I's Bluesky process and, also to be transparent to the instrument scientist who is encouraged to maintain the code of the instrument-specific part. A training course is provided to ensure the scientist is supported in this.

Assuming that detailed sample information is available SISSY I's (meta)data are considered to be rich enough to start the data conversion process automatically when a measurement ceases (see Fig. 3). The data conversion server works in the background, only accessing SISSY I's Mongo database once, in the beginning, to get the (meta)data of the run. The subsequent enrichment of (meta)data, the creation of the NeXus file and the ICAT ingest process doesn't affect the SISSY I instrument.

For samples not originating from the EMIL infrastructure there are two manual processes implemented on the data conversion server, shown in Fig. 3, which can be initiated by the instrument scientist of SISSY I. First, the single-step process provides a graphical user interface (GUI) through which essential metadata such as the proposal number, sample information and contacts to the experimental team can be provided. Then the experiment scientist starts the NeXus creation process which proceeds in the same way as the automatic workflow but adds the provided GUI metadata.

Second, for maximum control of ingested data a manual two-step process is offered. Here, the NeXus file is created in the beginning and the GUI displays the selected content that which can be adjusted to the experiment scientist's needs. Changes are written to the NeXus file which can be inspected by the favored software until the content meets the requirements.

In general, instrument scientists and data curators have to balance between automatic (meta)data processing offering low effort but bear the risk of wrong or missing (meta)data and a manual curation accompanied by increased workload but potentially more accurate information. Manual workflows would be the method of choice for example, when the instrument scientist must differentiate between data that can or cannot be made generally accessible, which is a concern when the beamline is shared between scientific users and users from industry whose results are confidential.

Currently, a coherent interface between various parts such as the e-logbook, repository, and the proposal system is missing and represent hurdles to the workflow, as additional user authentications are required. We are building bridges between services but an IT environment avoiding such problems would be desirable.

RESULTS AND DISCUSSION: FAIR ASSESSMENT

At SISSY I, we built a pure Python environment for beamline control, (meta)data acquisition, metadata enrichment, and conversion. This was made possible by the Python package Bluesky. We improve all fields of FAIR principles to increase data quality for long-term storage. To assess the FAIR status of the SISSY I instrument we apply the FAIR indicators of the FAIR maturity model [34] where results are summarized in table 1 following the presentation in [35]. Most of the time we restrict the discussion to indicators which are classified as 'essential' and therefore focus on the most important indicators to be fulfilled.

Reusability represents mainly the richness of metadata and, thus, we take measures on the instrument level to collect additional information. Since we were able to include sample information, electronic notebook and metadata identifiers such as ORCID or the instrument PID, we consider (meta)data to be sufficiently enriched that it can be reused. Moreover, it is presented in the machine-readable community standard NeXus. Currently, the method of including

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

FAIR	ID	Indicator	SISSY I
		Findability	
F1	RDA-F1-01M	Metadata is identified by a PID	3
F1	RDA-F1-01D	Data is identified by a PID	3
F1	RDA-F1-02M	Metadata is identified by a globally unique ID	3
F1	RDA-F1-02D	Data is identified by a globally unique ID	3
F2	RDA-F2-01M	Rich metadata is provided to allow discovery	4
F3	RDA-F3-01M	Metadata includes the identifier for the data	4
F4	RDA-F4-01M	Metadata can be harvested and indexed	3
		Accessibility	
A1	RDA-A1-01M	Metadata contains information to get access to the data	3
A1	RDA-A1-02M	Metadata can be accessed manually	4
A1	RDA-A1-02D	Data can be accessed manually	4
A1	RDA-A1-03M	Metadata identifier resolves to a metadata record	3
A1	RDA-A1-03D	Data identifier resolves to a digital object	3
A1	RDA-A1-04M	Metadata is accessed through standardized protocol	4
A1	RDA-A1-04D	Data is accessible through standardized protocol	4
A1	RDA-A1-05D	Data can be accessed automatically	3
A1.1	RDA-A1.1-01M	Metadata is accessible through a free access protocol	4
A1.1	RDA-A1.1-01D	Data is accessible through a free access protocol	4
A1.2	RDA-A1.2-01D	Access protocol supports authentication and authorization	4
A2	RDA-A2-01M	Metadata remains available after data is no longer available	4
		Interoperability	
I1	RDA-I1-01M	Metadata uses knowledge representation in standardized format	4
I1	RDA-I1-01D	Data uses knowledge representation in standardized format	4
I1	RDA-I1-02M	Metadata uses machine-understandable knowledge representation	4
I1	RDA-I1-02D	Data uses machine-understandable knowledge representation	4
I2	RDA-I2-01M	Metadata uses FAIR-compliant vocabularies	3
I2	RDA-I2-01D	Data uses FAIR-compliant vocabularies	3
I3	RDA-I3-01M	Metadata includes references to other metadata	4
I3	RDA-I3-01D	Data includes references to other data	4
I3	RDA-I3-02M	Metadata includes references to other data	2
I3	RDA-I3-02D	Data includes qualified references to other data	4
I3	RDA-I3-03M	Metadata includes qualified references to other metadata	4
I3	RDA-I3-04M	Metadata include qualified references to other data	2
		Reusability	
R1	RDA-R1-01M	Plurality of accurate and relevant attributes are provided	4
R1.1	RDA-R1.1-01M	Metadata includes information about the reuse license	3
R1.1	RDA-R1.1-02M	Metadata refers to a standard reuse license	3
R1.1	RDA-R1.1-03M	Metadata refers to a machine-understandable reuse license	3
R1.2	RDA-R1.2-01M	Metadata includes provenance information in community specific standard	4
R1.2	RDA-R1.2-02M	Metadata includes provenance information in cross community language	2
R1.3	RDA-R1.3-01M	Metadata complies with a community standard	4
R1.3	RDA-R1.3-01D	Data complies with a community standard	4
R1.3	RDA-R1.3-02M	Metadata is in machine-understandable community standard	4
R1.3	RDA-R1.3-02D	Data is in machine-understandable community standard	4

0 – not applicable, 1 – not being considered yet, 2 – under consideration, 3 – in implementation phase, 4 – fully implemented
Essential FAIR indicators are indicated by gray background.

Table 1: FAIR maturity model indicators for SISSY I in its current state of the commissioning phase.

the reuse license, in this case the HZB data policy, is under discussion.

Interoperability is closely connected to the way the knowledge is presented. Rather than the technical file format here the NeXus Definition Language NXDL is considered. Although no indicator is essential, NXDL is assumed to fulfill most requirements. However, to our understanding the NeXus vocabulary is not fully FAIR-compliant since the terms are not resolvable to let machines understand their meaning, but this work is up to the NIAC [36] which maintains the NeXus format. According to the assessment summarized in table 1 major deficiencies due to missing connection to other data, e.g., calibration measurements, must be addressed later in the commissioning phase.

By moving (meta)data to HZB's central repository ICAT we address the issue of accessibility. ICAT allows members of the experimental team to access (meta)data through a web interface. Missing the required identifier, (meta)data records

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

are not yet resolvable and, thus, access through machines is limited.

Like the accessibility, the findability depends mainly on features of the repository. To be globally visible (meta)data must be harvested by higher level services such as B2FIND and the repository itself must be known outside the HZB. A prerequisite to interact with global services is the use of globally unique PIDs that are automatically assigned to (meta)data and which will soon be implemented.

As a result, we expect to realize the essential FAIR indicators at the SISSY I end-station in the near future. However, please note that the assessment seems to be a rather subjective process since some FAIR indicators are ambiguous and need further clarification (e.g., the 'richness' of metadata).

CONCLUSION

We built a Python environment for automatic data workflow ranging from the sample handling to the instrument control software over the NeXus writing procedure to the repository ingest. Most of the required infrastructure is centralized to have a minimum impact on the instrument's work and share resources as e.g., the major part of writing NeXus files is independent of the scientific technique [15].

Important measures were implemented on the instrument level to increase the reusability significantly such as the automatic collection of sample information, and these have an impact on the workflow at the instrument. The early implementation of FAIR principles during the commissioning phase results in mutually induced workflow changes of instrument control and data management without running risk of increased downtime.

ACKNOWLEDGMENTS

The authors would like to thank Heike Görzig for constructive advice on software developments and Roland Mueller for helpful discussions.

This publication was supported by the Helmholtz Metadata Collaboration (HMC), an incubator-platform of the Helmholtz Association within the framework of the Information and Data Science strategic initiative.

GG, WS, OM conceptualized the project. GG and WS designed the workflow methodology and wrote software. WS and SV integrated Bluesky and metadata collection. GG and MK performed the investigation leading the FAIR assessment. MB and RW provided resources in the form of the EMIL laboratory and SISSY I end station. RK and NG provided software in the form of the e-logbook. RK worked on the instrument PID and provided the ICAT. GG wrote the original draft. All authors contributed to the review and editing process.

REFERENCES

[1] Mark D. Wilkinson, Michel Dumontier, and IJsbrand Jan Aalbersberg *et al*. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. https://doi.org/10.1038/sdata.2016.18.

[2] Klaus Lips, David Starr, and Marcus Bär *et al*. Emil: The energy materials in situ laboratory Berlin. *2014 IEEE 40th Photovoltaic Specialist Conference (PVSC)*, 11:0698–0700, 2014. https://doi.org/10.1109/PVSC.2014.6925017.

[3] FAIR Data Maturity Model Working Group. FAIR Data Maturity Model. Specification and Guidelines. *Zenodo*, 9:2651–2659, 2020. https://doi.org/10.15497/rda00050.

[4] Bluesky Project. https://blueskyproject.io/. Accessed: 2021-09-10.

[5] Markus Stocker, Louise Darroch, Rolf Krahl, Ted Habermann, Anusuriya Devaraju, Ulrich Schwardmann, Claudio D'Onofrio, and Ingemar Häggström. Persistent identification of instruments. *Data Science Journal*, 19(1):18, 2020. http://doi.org/10.5334/dsj-2020-018.

[6] Klaus Lips, Tim Schulze, and David Starr *et al*. EMIL: the energy materials in-situ laboratory Berlin – a novel characterization facility for photovoltaic and energy materials. *31st European Photovoltaic Solar Energy Conference and Exhibition*, 9:0698–0700, 2015. https://doi.org/10.4229/EUPVSEC20152015-1AO.2.1.

[7] Stefan Hendel, Franz Schäfers, Michael Hävecker, Gerd Reichardt, Michael Scheer, Johannes Bahrdt, and Klaus Lips. The EMIL project at BESSY II: Beamline design and performance. *AIP Conference Proceedings*, 1741:030038, 2016. https://doi.org/10.1063/1.4952861.

[8] Prevac. https://www.prevac.eu. Accessed: 2021-10-05.

[9] Rolf Follath, M. Hävecker, G. Reichardt, Klaus Lips, Johannes Bahrdt, Franz Schäfers, and P. Schmid. The energy materials in-situ laboratory Berlin (EMIL) at BESSY II. *Journal of Physics: Conference Series*, 425:212003, 2013. https://org.doi/10.1088/1742-6596/425/21/212003.

[10] EPICS. https://epics-controls.org/. Accessed: 2021-09-11.

[11] Ophyd. https://nsls-ii.github.io/ophyd/. Accessed: 2021-09-11.

[12] MongoDB. https://www.mongodb.com/en-us. Accessed: 2021-09-11.

[13] Will Smith *et al*. Status of Bluesky deployment at BESSY II. this conference.

[14] S. R. Heller, A. McNaught, and I. Pletnev *et al*. InChI, the IUPAC international chemical identifier. *J Cheminform*, 7:23, 2015. https://doi.org/10.1186/s13321-015-0068-4.

[15] R. Dimper, A. Götz, A. de Maria, V. A. Solé, M. Chaillet, and B. Lebayle. ESRF data policy, storage, and services. *Synchrotron Radiation News*, 32(3):7–12, 2019. https://doi.org/10.1080/08940886.2019.1608119.

[16] Joao Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. Understanding and improving the quality and reproducibility of Jupyter notebooks. *Empirical Software Engineering*, 26(4), 2021. https://doi.org/10.1007/s10664-021-09961-9.

[17] Declan Butler. Scientists: your number is up. *Nature*, 485:564, 2012. https://doi.org/10.1038/485564a.

[18] SISSY I endstation. https://doi.org/10.5442/NI000020.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[19] Soft x-ray EMIL beamline. https://doi.org/10.5442/NI000018.

[20] Hard x-ray EMIL beamline. https://doi.org/10.5442/NI000019.

[21] OASE – operando absorption and emission spectroscopy at EMIL. https://doi.org/10.5442/NI000021.

[22] SISSY-off SR. https://doi.org/10.5442/NI000022.

[23] Mark Könnecke, Frederick A. Akeroyd, Herbert J. Bernstein, Aaron S. Brewster, and Stuart I. Campbell *et al*. The nexus data format. *Journal of Applied Crystallography*, 48(1):301–305, 2015. https://doi.org/10.1107/S1600576714027575.

[24] Francesco De Carlo, Doğa Gürsoy, Federica Marone, Mark Rivers, and Dilworth Y. Parkinson. Scientific data exchange: a schema for HDF5-based storage of raw and analyzed data. *Journal of Synchrotron Radiation*, 21(6):1224–1230, 2014. https://doi.org/10.1107/S160057751401604X.

[25] U. K. Pedersen, N. Rees, M. Basham, and F. J. K. Ferner. Handling high data rate detectors at diamond light source. *Journal of Physics: Conference Series*, 425:062008, 2013. https://doi.org/10.1088/1742-6596/425/6/062008.

[26] N. P. Rees, M. Basham, F. J. K. Ferner, U. K. Pedersen, T. S. Richter, and J. A. Thompson. High speed detectors: problems and solutions. *Proceedings of ICALEPCS2013*, pages 1016–1019, 2013. paper WECOBA07, ISBN 978-3-95450-139-7.

[27] Mark Basham, Jacob Filik, Michael T. Wharmby, Peter C. Y. Chang, and Baha El Kassaby *et al*. Data Analysis WorkbeNch (DAWN). *Journal of Synchrotron Radiation*, 22(3):853–858, 2015. https://doi.org/10.1107/S1600577515002283.

[28] HZB Data Repository. https://data.helmholtz-berlin.de/. Accessed: 2021-09-11.

[29] Stephen M. Fisher, Frazer Barnsley, Wayne Chung, Sylvie da Graça Ramos, and Alex de Maria *et all*. The growth of the ICAT family. *NOBUGS 2016 Proceedings*, 9:2651–2659, 2008. https://doi.org/10.17199/NOBUGS2016.45.

[30] ICAT. https://icatproject.org/. Accessed: 2021-09-11.

[31] HZB Data Policy. https://hz-b.de/datapolicy. Accessed: 2021-09-11.

[32] B2FIND. http://b2find.eudat.eu/. Accessed: 2021-09-11.

[33] R. Müller, A. Balzer, P. Baumgärtel, O.-P. Sauer, and G. Hartmann *et al*. Modernization of experimental data taking at BESSY II. *Proceedings of ICALEPCS2019*, pages 65–69, 2019. https://doi.org/10.18429/JACoW-ICALEPCS2019-MOCPL02.

[34] FAIR Data Maturity Model. Specification and Guidelines. https://doi.org/10.15497/rda00050. Accessed: 2021-09-11.

[35] Christophe Bahim, Carlos Casorrán-Amilburu, Makx Dekkers, Edit Herczog, and Nicolas Loozen *et all*. The FAIR data maturity model: An approach to harmonise FAIR assessments. *Data Science Journal*, 19(41):1–7, 2020. https://doi.org/10.5334/dsj-2020-041.

[36] NeXus International Advisory Committee. https://www.nexusformat.org/NIAC.html. Accessed: 2021-09-11.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR01

RomLibEmu: NETWORK INTERFACE STRESS TESTS FOR THE CERN RADIATION MONITORING ELECTRONICS (CROME)

K. Ceesay-Seitz†, H. Boukabache*, M. Leveneur, D. Perrin, CERN, Geneva, Switzerland

Abstract

The CERN RadiatiOn Monitoring Electronics (CROME) are a modular safety system for radiation monitoring that is remotely configurable through a supervisory system via a custom protocol on top of a TCP/IP connection. The configuration parameters influence the safety decisions taken by the system. An independent test library has been developed in Python in order to test the system's reaction to misconfigurations. It is used to stress test the application's network interface and the robustness of the software. The library is capable of creating packets with default values, autocompleting packets according to the protocol and it allows the construction of packets from raw data. Malformed packets can be intentionally crafted and the response of the application under test is checked for protocol conformance. New test cases can be added to the test case dictionary. Each time before a new version of the communication library is released, the Python test library is used for regression testing. The current test suite consists of 251 automated test cases. Many application bugs could be found and solved, which improved the reliability and availability of the system.

INTRODUCTION

The Radiation Protection group within CERN is responsible for measuring levels of ionizing radiation at the CERN sites, experimental areas and in service caverns besides the LHC experiments in order to ensure the radiological safety of the persons on the CERN site as well as the people living in its neighbourhood. The CERN RadiatiOn Monitoring Electronics (CROME) have been developed in-house with the purpose of replacing the older radiation monitoring systems ARCON and RAMSES. In contrast to the old systems, CROME consists of fully independent units, called CROME Measurement and Processing Units (CMPUs), which perform their safety functions autonomously [1]. A CMPU consists of a radiation detector, which is an ionization chamber in most cases, a front-end board for the readout and the processing unit. The latter has at its core a Zynq-7000 System-on-Chip (SoC) with an embedded Linux running on the Processing System (PS) with integrated 32-bit ARM cores and a Programmable Logic (PL) section. Because the PL can operate autonomously and the design architecture is more immune to higher radiation than the PS [2], all the safety critical calculations and decision making are implemented inside the PL.

The CMPU's functionality is entirely configurable at runtime with roughly 150 parameters. This has the advantage that CROME can be deployed for very different usage scenarios – e.g. in service caverns or experimental areas with higher levels of radiation where it can be configured to trigger visible and audible alarms and with the possibility of being connected to machine interlocking systems, or as environmental monitors where the natural background radiation is monitored over long periods for informational purposes.

Figure 1 presents a system overview. During operation the CMPUs are connected to a SCADA supervisory system called REMUS – Radiation and Environment Monitoring Unified Supervision [3]. The CMPUs on the one communication end and the REMUS servers on the other end use the ROMULUS library [4] for communicating with each other. The library implements a custom protocol on top of TCP/IP that can be used to remotely configure the parameters on the CMPU at runtime, to read out its current and historical status, and to receive real time as well as historical measurement values.

The remote parametrization via REMUS is the only dedicated mechanism for users to configure the behaviour of the CROME system during operation. Radiation protection experts use REMUS to configure the CMPUs corresponding to the expected radiation conditions and safety requirements of a zone. Operators in the control room use REMUS to monitor the status and measurement results sent by the CMPUs.

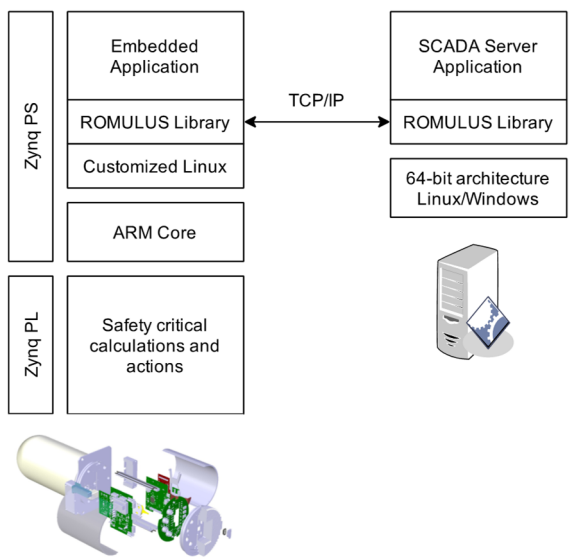


Figure 1: CROME Overview.

Since it influences the functionality as well as the safety function of the system, the communication mechanism needs to be robust and reliable. The RomLibEmu, the ROMULUS Library Emulation and Test Tool, has been developed to test the functionality and robustness of the ROMULUS library as well as of the CMPU's application.

† katharina.ceesay-seitz@cern.ch
* hamza.boukabache@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The paper is structured as follows. First a definition for robust systems and robustness testing is provided. Then the RomLibEmu's concepts and functionalities are described in detail. Finally, the results and benefits obtained from using the tool are highlighted. A summary and an outlook conclude the paper.

BACKGROUND AND RELATED WORK

A system is considered as robust when it shows stability and acceptable behaviour in unforeseen operating conditions. Furthermore, it must not accept invalid input and certainly must not produce faulty output if presented with unexpected inputs. Rather it has to either reject the input or put itself into a safe or degraded state from which it can recover. Robustness is defined by the 610.12-1990 IEEE standard glossary of software engineering as "the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions". The latter could be network traffic overload, attacks or a misbehaving connected system [5]. It could also be physical conditions such as extreme temperatures or high radiation levels.

Robustness tests complement functional tests with test cases that particularly focus on sending boundary values or unexpected input values to a system with the goal of determining whether the system enters into an unwanted state such as an abort or a non-responsive state[6]. Common techniques for robustness testing are model-based techniques, fault injection, fuzzing, interception, code changes injection and mutation testing. Communication protocols are often tested with model-based techniques where a model is built from the specification which is then either directly evaluated or used to generate test cases. The most commonly used technique to evaluate the robustness of embedded systems is fault injection [7]. RomLibEmu currently focuses mainly on fault injection. In future it might be extended with model-based or fuzzing approaches.

The CRASH scale has been developed for classifying the severity of failures of operating systems when tested for robustness. CRASH stands for Catastrophic, Restart, Abort, Silent, Hindering [8]. It is widely used to classify robustness test results [7]. It was also adapted to classify failures of controllers in self-adaptive systems. Similar to our results the most common failure of the tested controller fell into the "Silent" category [9].

Besides aiding in the classification of test results it also provides a good guidance for scenarios that have to be tested for. We reinterpreted the scale as Catastrophic, Restart, Abort, Silent/Safety, Hindering/Harmless.

RomLibEmu: ROMULUS LIBRARY EMULATION AND TEST TOOL

Purpose and Motivation

As mentioned in the introduction, the network interface is the only dedicated mechanism for configuring the CROME Measurement and Processing Units (CMPUs) during operation. The first parametrization can also be done via the local file system, but this interface is only accessible to the development and maintenance team of CROME. Further utility tools exist that have been developed for testing and diagnosis. A physical test case, called CROME Case, that can read and manipulate external interfaces of the CMPU, has been developed as well. The tools and the test kit use the ROMULUS library for communication via the network. The library is intended to construct well-formed messages conforming to the ROMULUS-REMUS communication protocol. It only allows to send values of the expected datatypes for parameters and it is not capable of intentionally crafting malformed packets. While this feature strengthens the communication mechanism when the same library is used on both communication end points, it prevents the usage of the library itself for testing its own robustness and that of the CMPU application. The RomLibEmu, the ROMULUS Library Emulation and Test Tool, has been developed to overcome this limitation.

There are several scenarios in which it could happen that unexpected messages or malformed packets are sent to a CMPU:

- A fault in the ROMULUS library.
- A fault in the application that uses the ROMULUS library.
- The ROMULUS library versions of the CMPU application and the REMUS server are incompatible.
- Bad network connection or unexpected termination of the remote application causes ROMULUS packets to arrive only partially.
- Network or user-level misconfiguration:
  Packets that were not intended to be sent to the CMPU reach it because of a network misconfiguration. The application tries to interpret the irrelevant data according to the ROMULUS-REMUS communication protocol.
- Intentional attacks against CROME

Robustness testing of the CMPU's network interface is crucial to increase the availability and reliability of the radiation monitors. The availability would be affected by application crashes or aborts due to unexpected network traffic or heavy loads. The reliability would be affected if the CMPU would accept input values which it cannot process correctly. That could be e.g. input values that cause internal overflows or divisions by zero. The expected functionality of the CMPU would be affected if it would accept parameter values outside of the ranges defined in the requirements. These functional tests could be implemented by using the ROMULUS library itself. The already existing tools were mainly intended for manual tests and diagnosis. The RomLibEmu additionally facilitates the creation and generation of automated functional and robustness test suites. They can be used for automated regression testing as well.

Capabilities

The RomLibEmu has been developed independently from the ROMULUS library based on the protocol specifi-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR01

cation. Figure 2 shows an overview of the test setup. The tool can be used to test any application software that uses the ROMULUS C library or any other library or application that implements the ROMULUS-REMUS communication protocol. The library provides functions for creating, sending and receiving packets via TCP/IP. Packets can be autocompleted to conform to the ROMULUS protocol or they can be intentionally crafted to not or only partially conform to the protocol. Further functions can interpret received packets and check for unexpected responses.

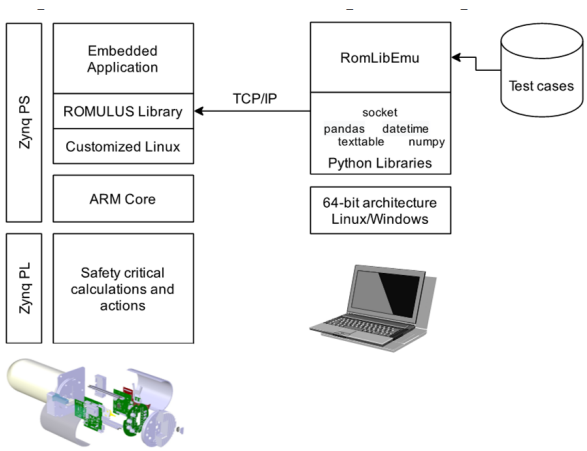


Figure 2: RomLibEmu Overview.

Metadata The protocol information is extracted from the protocol definition document in a semi-automated manner. The protocol consists of several different commands with similar structures. Each of them is identified by a unique command code and has a corresponding response code. Depending on the command type either the command or the response contains a list of parameters with or without added ID and of specified data types. Each packet further contains the data length, number of parameters, an ID and a checksum [4]. The command and response codes as well as the acknowledgement code for the messages have to be manually extracted from the documentation into a Python dictionary. The document contains several tables that define the parameter names, IDs and data types for the different messages. These need to be copied into a CSV (comma-separated values) file. The metadata generation scripts of the RomLibEmu reads the information and generates metadata dictionaries which are stored into dedicated files. These dictionaries can be used to access the parameters by name or ID and retrieve the corresponding data type byte size. The generated files are included into the RomLibEmu library. In case different protocol versions shall be tested, it is possible to use different input folders and include different dictionaries into the library.

Test case generation – conforming packets The library can be used to create test cases either manually or automatically. A test case can be specified as a dictionary with the ROMULUS packet field names as keys and the field's data as values. All values need to be provided as bytes in the correct byte order as described by the protocol. Values like the command codes can be retrieved from the generated dictionaries by their names. Several utility functions are available to facilitate the test case construction in a human readable form. The "DATA" field expects as value a dictionary that contains key-value pairs with the name of a parameter as key and its data as value. The parameter ID that will be placed into the protocol message is automatically determined from the metadata. Since the ID depends on the command type (there could be multiple parameters with the same name but different ID for different command types), a command code needs to be specified in the test case. In order to create a conforming packet, the ID, command code, number of parameters and the data need to be specified. The RomLibEmu then calculates the data length and the checksum and places them into the ROMULUS packet.

A test case also contains the expected response by the CMPU under test. The expected acknowledgement code depends on the test scenario and must therefore be added manually. In most cases the expected data values are only known for parameters and not for sensor measurements like the radiation dose or temperatures. In some test cases they can be added to the expected response packet. It is possible to add place holders for data values that cannot be automatically checked. Several fields of the expected response can be automatically determined by the RomLibEmu. The sender ID, the command's response code, the expected number of parameters and the data length can be generated. The checksum of the response packet can be calculated only if the expected data was known.

Test case generation – non-conforming packets Raw data can be inserted by adding a key that starts with the keyword "raw" and any data as value. These fields can be used to craft invalid packets. The default value of each field is an empty string. It is the responsibility of the test case writer to decide whether the lengths of the fields should match the ROMULUS-REMUS communication protocol for conforming packets or whether a non-conforming packet shall be created. In order to construct a packet manually it is possible to add the whole packet as a byte string to one of the fields of the packet and leave all other fields empty. Packets can also be modified after auto completion to facilitate the generation of malformed packets.

Use case The main purpose of the RomLibEmu is to facilitate the creation of automated test suits. Test cases can be defined in a test case dictionary. The RomLibEmu provides a function that traverses this dictionary and sends each packet over a TCP/IP socket to the target device. After sending a packet it waits for the response and compares it with the expected one that was defined in the test case. This verifies the ROMULUS packet structure which tests the ROMULUS library implementation as well as the content of the packet which tests the software application that uses the library. Automated test suits can be used fully or partially in regression test runs which are performed whenever a new software version is available.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The functions of the RomLibEmu can also be used to freely create more complex functional test cases that consist of several ROMULUS commands. Scenarios that already lead to failures of the system can be reconstructed for analysis. Once the underlying fault has been removed, the same test case can be used in regression runs. The library can also be used by developers for test-driven development.

RESULTS

A test plan has been written containing around 80 main test cases which consist of several sub test cases. They are associated to the following categories:

1. Good cases – Packets that conform to the ROMULUS-REMUS communication protocol.
2. Application tests – Packets with correct ROMULUS packet structure, but bad parameter configuration.
3. Bad network connection – Simulating the loss of packets, long delay, etc.
4. ROMULUS library tests – Malformed ROMULUS packets that were generated due to bugs in the ROMULUS library; Network packets that were not targeted to the CMPU but arrive because of misconfiguration and are therefore treated by the application like legitimate traffic.
5. Denial-of-Service – Overload of the network interface by sending too many messages in a short time or in parallel, unexpectedly long packets.
6. Intentional attacks
7. Regression tests – Test cases that have already revealed a bug in some version of the application or the ROMULUS library

The responses of the CMPU's application can be classified according to the CRASH scale [8] as follows:

- Catastrophic – The OS crashes.
- Restart – Application hangs and requires a restart.
- Abort – The application crashes.
- Silent/Safety – An application error is expected but does not get triggered.
- Hindering/Harmless – The application returns a wrong error code.

Each time before a new software or firmware version is released, the test suite implemented with the RomLibEmu is executed. Table 1 lists the failures of the test runs per CRASH category. The robustness and in particular the availability of the CMPU was tested by trying to cause a "Catastrophic", "Restart" or "Abort" scenario. No test scenario fell into the "Catastrophic" category. One test case caused the application to hang and four test cases caused the application to crash, falling into the "Abort" category. "Silent" faults are very critical for safety, therefore we renamed the category into "Silent/Safety". These are scenarios in which the application would accept bad input that it cannot handle and it would not notify the user. In such scenarios one would rely on a faulty safety system in the belief that it is operating as specified, which could be very dangerous. The faults falling into the "Hindering/Harmless" category were not considered as critical. They are wrong behaviour from a functional perspective, but from a safety perspective in the case of the CMPU it is sufficient if any error code is returned and bad input is rejected. For another system, however, these faults may be critical. We also added a category "Functional" which contains test cases that failed because the response of the system did not fully conform to the protocol, but the error code was the expected one. From safety perspective they are not critical either.

The following test cases lead to "Abort" responses, meaning that the application crashed. (The numbers in brackets following each paragraph are the test case categories):

- A floating point parameter was set to quiet NaN (Not a Number).
  The ROMULUS library is not supposed to accept NaNs, therefore this tested the library implementation as well. Interestingly a test case with a signalling NaN passed. (2, 4, 6)
- An Ethernet frame with maximum data length was sent. The length of the IP header plus the length of the TCP header plus the length of the data equalled the Ethernet MTU of 1500. The TCP data was all 0. A similar test case where the TCP data contained arbitrary data, but the data length specified in the packet matched the maximum legal data length and the ROMULUS packet checksum was correct passed. (4, 6, 7)
- A packet with arbitrary data in the TCP data section with a packet length larger than the maximum ROMULUS packet length was sent. The data length inside the packet was set to 0 and the packet had a wrong checksum. (4, 6, 7)
- Many commands were sent in parallel. (2, 4, 5, 6)

Some examples of "Silent/Safety" failures were:

- Negative/positive infinites as well as quiet and signalling NaNs were sent to floating point parameters. The values were accepted and passed on to the PL that contains the safety critical code. (2, 4, 6)
- Invalid packet structures were accepted. The application therefore treated frame data like the checksum as parameter data and stored it silently. (2, 4, 6)

Table 2 summarizes the test runs. With each run some faults were removed and some more test cases were added. The number of passing test cases increased with each improved software version. In some intermediate test runs, however, previously passing test cases failed due to regressions that were introduced into the software or the firmware of the PL. Thanks to the test suite this could be spotted and fixed before releasing the new software and firmware versions.

CONCLUSION AND OUTLOOK

The RomLibEmu tool has been developed to test the robustness of the ROMULUS library as well as the CMPU's application that uses the library. It was successfully used

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR01

Table 1: Failures per CRASH Category

Test Run	Date of SW	Silent/ Safety	Abort	Restart	Hindering	Functional
1	04/09/19	19	2	1	3	2
2	11/12/19	10	4	0	3	2
3	20/12/19	6	0	0	3	2
4	26/05/20	6	0	0	3	3
5	02/03/21	2	0	0	3	2
6	30/09/21	0	0	0	3	0

Table 2: Test Run Summary

Test Run	Date of SW	Nr. Test Cases	Passed	Failed
1	04/09/19	57	30	27
2	11/12/19	58	39	19
3	20/12/19	64	53	11
4	26/05/20	65	53	12
5	02/03/21	81	74	7
6	30/09/21	83	80	3

Many of the main test cases consist of several sub cases. In total the current test suite contains 251 test cases.

for finding faults triggered by unexpected parameter values or malformed ROMULUS packets. It has been further used to test the input sanitizing module which performs parameter range checks once when receiving parameters over the network or reading them from the file system as well as once before writing derived values to the Programmable Logic section of the SoC. A few test cases have already been written that test functionalities of the full system covering PS and PL sections of the SoC.

This test bench clearly improved the quality, availability and safety of the CROME Measuring and Processing Units. So far the test bench focuses only on a small section of the functionality and on the communication interface. In future version we intend to extend the tool to further facilitate the test case generation for functional and safety tests. More benefits are expected from a more comprehensive test suite. There is also room for more automation. Model-based testing may be done by adding the capability of generating random input values and by adding an interface to a reference model of the system that predicts the expected outputs. Furthermore the tool may be combined with fuzz testing tools or modules with focus on network security.

REFERENCES

[1] H. Boukabache, M. Pangallo, G. Ducos, N. Cardines, A. Bellotta, C. Toner, D. Perrin, and D. Forkel-Wirth, "TOWARDS A NOVEL MODULAR ARCHITECTURE FOR CERN RADIATION MONITORING", *Radiation Protection Dosimetry*, vol. 173, April 2017, pp. 240–244. doi:10.1093/rpd/ncw308

[2] C. Toner, H. Boukabache, G. Ducos, M. Pangallo, S. Danzeca, M. Widorski, S. Roesler, and D. Perrin, "Fault resilient FPGA design for 28 nm ZYNQ system-on-chip based radiation monitoring system at CERN", in *Microelectronics Reliability*, vols. 100–101, p. 113492, 2019. doi:10.1016/j.microrel.2019.113492

[3] A. Ledeul, G. Segura Millan, A. Savulescu, B. Styczen, and D. Vasques Ribeira, "CERN Supervision, Control and Data Acquisition System for Radiation and Environmental Protection", in *Proc. 12th International Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'18)*, Hsinchu City, Taiwan, Rep. of China, 2018, pp. 248-252. doi:0.18429/JACoW-PCaPAC2018-FRCC3

[4] A. Yadav, H. Boukabache, K. Ceesay-Seitz, and D. Perrin, "ROMULUSlib: An autonomous, TCP/IP-based, multi-architecture C networking library for DAQ and Control applications", presented at 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOBR01, this conference.

[5] S. Shah, D. Sundmark, B. Lindström, and S. Andler, "Robustness Testing of Embedded Software Systems: An Industrial Interview Study", in *IEEE Access*, vol. 4, pp. 1859-1871, 2016. doi:10.1109/ACCESS.2016.2544951

[6] C. Hutchison *et al.*, "Robustness Testing of Autonomy Software", *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2018, pp. 276-285. doi:10.1145/3183519.3183534

[7] N. Laranjeiro, J. Agnelo, and J. Bernardino, "A Systematic Review on Software Robustness Assessment", *ACM Computing Surveys*, vol. 54, issue 4, nr. 89, pp. 1-65, 2021. doi:10.1145/3448977

[8] P. Koopman, J. Sung, C. Dingman, D. Siewiorek, and T. Marz, "Comparing operating systems using robustness benchmarks", *Proceedings of SRDS'97: 16th IEEE Symposium on Reliable Distributed Systems*, 1997, pp. 72-79. doi:10.1109/RELDIS.1997.632800

[9] J. Camara, R. de Lemos, N. Laranjeiro, R. Ventura, and M. Vieira, "Robustness Evaluation of Controllers in Self-Adaptive Software Systems", *2013 Sixth Latin-American Symposium on Dependable Computing*, 2013, pp. 1-10. doi:10.1109/LADC.2013.17

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

TOWARDS THE OPTIMIZATION OF THE SAFETY LIFE-CYCLE FOR SAFETY INSTRUMENTED SYSTEMS

B. Fernández*, G. De Assis, R. Speroni, T. Otto, E. Blanco,
CERN, Geneva, Switzerland

Abstract

The design and development of Safety Instrumented Systems (SIS) according to the IEC 61511 standard is a long and costly process. Although the standard gives recommendations and guidelines for each phase of the safety life-cycle, implementing them is not a simple task.

Access to reliability data, hardware and systematic safety integrity analysis, software verification, generation of reports, guarantee of traceability between all the phases and management of the project are some of the main challenges. In addition, some of the industrial processes or test benches of large scientific installations are in continuous evolution and changes are very common. This adds extra complexity to the management of these projects.

This paper presents an analysis of the safety life-cycle workflow and discusses the biggest challenges based on our experience at CERN. It also establishes the basis for a selection of the tools for some of the safety life-cycle phases, proposes report templates and management procedures and, finally, describes the roles of the different members in our functional safety projects.

INTRODUCTION

The design, development and maintenance of Safety Instrumented Systems (SIS) requires a lot of resources and time for a company or organization. It is not enough to develop a reliable SIS based on good engineering practices, it is necessary to prove that the Safety Instrumented Functions (SIF) reduce the existing risk to the tolerable region. The functional safety standards provide the guidelines to design and develop such systems and the methods to prove the compliance with the risk reduction target. For industrial processes, the IEC 61511 [1] is the most appropriate standard. It uses the same principles as the IEC 61508 standard with a more specific language and context.

The IEC 61511 Safety Life-Cycle

Figure 1 shows the so-called IEC 61511 safety life-cycle, whose requirements are specified in the Clause 6 of the IEC 61511-1:2016. This Clause defines the different phases, organizes the technical activities and ensures that adequate planning exists for the development of the SIS.

In order to claim conformance with the IEC 61511 standard, all requirements from Clause 5 to Clause 19 from the 61511-1:2016 have to be met and the corresponding reports have to be created. All these requirements are clearly specified, but how to implement them is the real challenge.

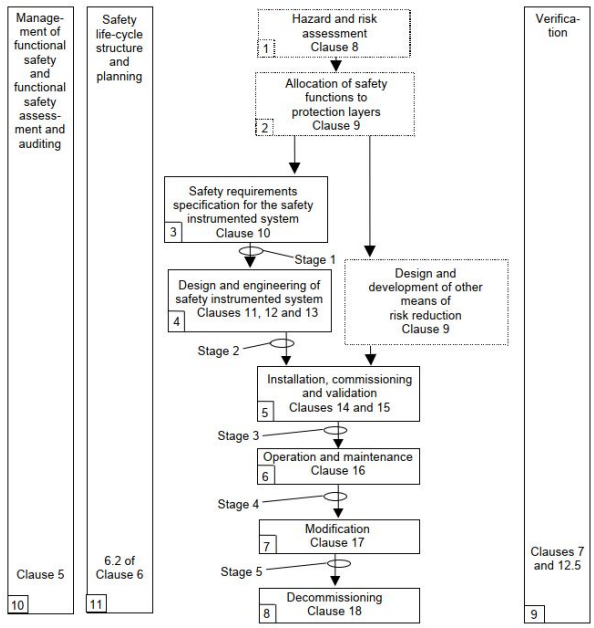


Figure 1: IEC 61511 safety life-cycle.

Challenges

The main challenges related to the implementation of the safety life-cycle are common to most industries. In our experience, proving the compliance with the standard and guaranteeing the traceability between all phases of the safety life-cycle are two of the most critical ones. Compliance is a very costly and time-consuming process, and lack of traceability most certainly would create discrepancies in the project documents, delays and potential errors.

In addition, at CERN (European Organization for Nuclear Research) and most probably in other large scientific installations, some of the industrial processes or test benches are in continuous evolution and changes that influence the safety of the installation are very common.

Objectives

Our goal is to overcome the technical and organizational challenges to optimize the allocated resources for the design, development and maintenance of SIS.

More specifically, we aim to:

- Create report templates that are necessary for the documentation and management of such projects.
- Reuse and integrate the existing tools we have at CERN that can be applied to any of the phases of the safety life-cycle.
- Discuss the management procedures and the roles of the different members in our functional safety projects.

* borja.fernandez.adiego@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

COMMERCIAL TOOLS

There are many commercial tools that offer solutions for one or several phases of the safety life-cycle. Table 1 presents some of the most relevant ones.

Exida's toolkit, exSILentia®, is one of the most popular tools. It covers all the phases of the life-cycle and certainly fulfils the requirements of most engineering teams to design and develop SIS in terms of reporting, management, traceability and reliability. This tool could certainly bring a lot of benefits to our workflow, especially traceability. There are however a few requirements that are not covered by this framework, to the best of our knowledge, regarding code generation and formal verification. At CERN, we have adopted alternative solutions, which are discussed in the following sections.

Another relevant tool is the Reliability Workbench provided by Isograph. It allows the creation of reliability models, such as FTA (Fault Tree Analysis) and RBD (Reliability Block Diagrams), and analyses their compliance with the IEC 61508 standard [2] in terms of random failures and architectural constraints. An example can be found in [3].

The rest of the paper analyses some of the most challenging phases of the safety life-cycle based on our experience and describes the adopted solutions.

HAZARD AND RISK ASSESSMENT, ALLOCATION OF SAFETY FUNCTIONS TO PROTECTION LAYERS AND SAFETY REQUIREMENTS SPECIFICATION

The hazard and risk assessment, allocation of safety functions to protection layers and Safety Requirements Specification are the first three phases of the safety life-cycle. The IEC 61511-1:2016 Clause 8 defines the requirements to perform

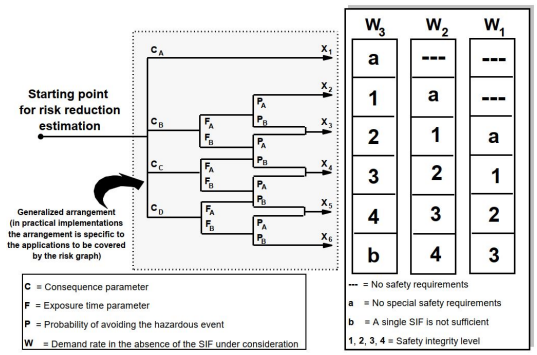


Figure 2: Risk graph: general scheme from IEC 61511-3:2016 Annex D.

the hazard and risk (H&R) analysis from the process and the BPCS (Basic Process Control System). Clause 9 defines the requirements for the allocation of safety functions to protection layers. Clause 10 provides all the requirements to produce the SRS (Safety Requirements Specification) document.

The objectives of these phases are: (1) identify the hazards and risks of the industrial process and the BPCS, and evaluate the necessary risk reduction to the tolerable levels, (2) propose a risk mitigation strategy and (3) provide a detailed and unambiguous specification of the SIS. Annex A from IEC 61511-3:2016 describes the concepts of "tolerable risk target", the layers of protection and how to select the appropriate method to evaluate the risks.

In our case, in order to identify the dangerous hazards that can provoke a risk to operators, environment or asset loss (damage to the installations), we usually apply the FMEA (Failure Mode and Effect Analysis) method. To quantify the necessary risk reduction for each failure mode, we use

Table 1: Safety Life-cycle Tools and Software Suites.

Tool	Safety life-cycle coverage	Reference
exSILentia® (Exida)	All phases	https://www.exida.com/
Safeguard Profiler	Phases 1, 2, 3, 4 and 6 (Bowtie, LOPA analysis, SRS, SIS design, SIL verification, Proof test analysis)	https://www.acm.ca/safeguard-profiler/
SISsuite	All phases	https://www.sissuite.com/
SLM V2	All phases	https://mangansoftware.com/slm-v2/
Vertigo™	Phases 3 and 4 (Equipment failure rate database, SRS and SIL verification)	https://www.kenexis.com/software/sis-lifecycle-management-and-sil-verification/
SIL Solver®	Phase 4 (SIL verification)	https://sis-tech.com/applications/sil-solver/
Isograph's Reliability Workbench	Phase 4 (SIL verification)	https://www.isograph.com/software/reliability-workbench/
Siemens Safety Matrix Engineering Tool	All phases	https://assets.new.siemens.com/siemens/assets/api/uuid:f18dcad1-9faf-4f33-8c20-c8390d176993/safetymatrixflyerfinal-300.pdf
SILcet	Phase 4 (SIL verification)	https://safetyandsis.com/sil-verification/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 2: Simplified FMEA Example

Subsystem	Failure Mode	Effects	Causes	Current mitigation measures
Water-cooled system	High temperature	Melting insulation, short circuit and electrocution	Water leak	None

Table 3: Example of Risk Evaluation for Personnel Using the Risk Graph Method from IEC 61511-3:2016 - Annex D

Consequence	Occupancy (Frequency of exposition	Possibility of avoidance	Base prob. of failure	SIL target
CC (1 fatality is possible)	FA (less than 10% of working time)	PB (no other system)	W1 (less than once every 10 years)	SIL1 (risk reduction bet. 10 and 100)

one of the methods proposed by the IEC 61511-3:2016, the calibrated risk graph (Annex D). Figure 2 shows the general scheme of this method.

Tables 2, 3 and 4 show a simplified example of the selected methods (FMEA and calibrated risk graph) applied to an industrial installation at CERN.

In Table 2, a failure mode, its effects and causes are described for a simple example of water-cooled cable subsystem, as part of a superconducting magnet test bench facility.

In Table 3, the calibrated risk graph method is applied to evaluate the necessary risk reduction to protect the operators of the installation for the failure mode example shown in Table 2. The risk reduction factor (RRF) is given by the target SIL (Safety Integrity Level). In this example, a SIL1 implies a RRF between 10 and 100.

In Table 4, the same method is applied for the same failure mode but evaluating the asset loss. In this case, the failure mode could provoke a serious damage on the water-cooled cable subsystem and weeks of delay in the test program.

In our experience, the biggest challenge in these two phases is the risk graph calibration for asset loss. Calibration means assigning numerical values to the risk graph parameters in accordance with the corporate risk criteria. The calibration to define the tolerable risk level for personnel protection is relatively straight forward, following the examples given by the standard (e.g. table D.2 from the IEC 61511-3:2016 Annex D, or table E.1 from the IEC 61508-5:2016 Annex E). A bigger challenge is to calibrate the graph for asset loss. A failure in one of our industrial processes would normally provoke a delay in the physics program of one of our particle accelerators (e.g. the Large Hadron Collider) or in the test program of one of our test benches. We have calibrated the consequences in "delay time", which is a way for our process and accelerator experts to define the tolerable risk. It is, however, very hard to generalize and provide the same calibration for all the industrial processes we have at CERN. In Table 5, we show an example of the calibration for asset loss in one of our projects.

This calibration has a very big impact in the next phases of the project. It was a very time-consuming process, since it needed the consensus and approval of many members of the functional safety project. However, we have built a sound base for future projects.

Once the risks are identified and the necessary risk reductions (SIL) are calculated, the functional safety expert must propose a mitigation strategy for this risk. If the mitigation method is a SIF, the rest of the safety life-cycle should be completed and a detailed SRS document must be provided to proceed to the next phase. However, SIF is not the only option to mitigate a risk. Other protection layers that should be considered to mitigate a risk (see IEC 61511-1 Clause 9).

This exercise was performed by the functional safety engineer, the process expert, the automation engineer and the Departmental Safety Officer (in charge of evaluating and fixing the acceptable risk levels for the project). Report templates were created for the FMEA-based risk analysis, the risk assessment and the SRS.

SIS DESIGN AND ENGINEERING

The IEC 61511-1:2016 Clauses 11, 12 and 13 define the requirements of the phase 4 of the safety life-cycle: Design and engineering of the SIS.

The goal is to design an SIS compliant with the requirements defined in the SRS document. This phase is a very

Table 4: Example of Risk Evaluation for Asset Loss Using the Risk Graph Method from IEC 61511-3:2016 - Annex D

Consequence	Occupancy (Frequency of exposition	Possibility of avoidance	Base prob. of failure	SIL target
CC (several weeks of delay)	FB (max. occupancy)	PB (no other system)	W1 (less than once every 10 years)	SIL2 (risk reduction bet. 100 and 1000)

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

Table 5: Example of Risk Graph Calibration for Asset Loss

Consequence		Occupancy		Possib. of avoidance		Prob. of failure	
CA	delay < few hours	FB	always	PA	automatic system that detects and alerts the operators	W1	< 1 failure per 10 years
CB	few hours < delay < few days			PB	There is not	W2	< 1 failure per year
CC	few days < delay < few weeks					W3	> 1 failure per year
CD	delay > a month or cancellation of test program						

long and time consuming process. For each SIF specified in the SRS, four main analyses must be conducted: (1) the random hardware failures, (2) the architectural constraints, (3) the systematic failures or selection of the devices and (4) the application program.

SIF Random Hardware Failures and Architectural Constraints Analysis

In this analysis, the functional safety engineer performs the calculation of the probability of dangerous failure on demand (PFD_{avg}), when the SIF operation mode is low demand; or the probability of dangerous failure per hour (PFH), when the operation mode is high or continuous demand. For this purpose, the reliability data of the SIF components is needed (e.g. failure rate, mean time to failure, etc.). Once the data for the individual components (sensors, final elements and controller) is collected, a reliability model of the SIF must be created and the global PFD_{avg} or PFH is calculated. Table 6 shows the random hardware failure requirements in demand mode for each SIL.

Table 6: Safety Integrity Requirements for Random Hardware Failures from IEC 61511-1:2016 Clause 9.2.3

Demand Mode of Operation		
Safety Integrity Level (SIL)	PFD_{avg}	**Required risk reduction**
4	$\geq 10^{-5}$ to $< 10^{-4}$	$> 10^{4}$ to $\leq 10^{5}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$> 10^{3}$ to $\leq 10^{4}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$> 10^{2}$ to $\leq 10^{3}$
1	$\geq 10^{-2}$ to $< 10^{-1}$	$> 10^{1}$ to $\leq 10^{2}$

However, this analysis is not enough. It is also needed to evaluate the architecture of the SIF. For sensors and final elements, the IEC 61511-1:2016 Clause 11.4 provides the requirements for the minimum hardware fault tolerance (HFT) of each element of the SIF. Table 7 shows the HFT requirements for each SIL and operation mode.

Although these requirements can be relaxed if the parameter Safe Failure Fraction (SFF) is known for each element. In this case, it is possible to use the Route 1_H method provided by the IEC 61508-2:2010 Clause 7.4.4. Table 8 shows the HFT requirements for each SIL and the SFF ranges for type A elements (typically, devices without any processor, e.g. simple mechanical sensors or final elements). A similar table with more strict requirements for type B elements (e.g. controllers) can be found in Clause 7.4.4.

Table 7: Minimun HFT Requirements According to SIL from IEC 61511-1 Clause 11.4

SIL	Minimun HFT
1 (any mode)	0
2 (low demand mode)	0
2 (continuous mode)	1
3 (high demand mode) or continuous mode)	1
4 (any mode)	2

Table 8: Maximum Allowable SIL for a Safety Function Carried Out by a Type A Safety-Related Element or Subsystem from IEC 61508-2:2010 Clause 7.4.4

SFF	HFT		
	0	1	2
$SFF < 60\%$	SIL1	SIL2	SIL3
$60\% \leq SFF < 90\%$	SIL2	SIL3	SIL4
$90\% \leq SFF < 99\%$	SIL3	SIL4	SIL4
$SFF \geq 99\%$	SIL3	SIL4	SIL4

In our case, Isograph is the selected tool for checking the architectural constraints and quantifying the random hardware failures. This tool is widely used at CERN for reliability calculations. It provides access to the reliability component data and supports the IEC 61508 standard (PFD_{avg} and PFH formulas are included, as well as the HFT tables from Route 1_H). The functional safety engineer can create the failure models, which allows to calculate the random hardware failures according to this standard. Figure 3 shows a diagram of a Fault Tree that models one of our SIFs within the Isograph's Reliability Workbench. The SIF components, to mitigate the risk from Table 2, are: a thermoswitch, the Programmable Logic Controller (PLC) and its input/output cards, and the final elements (in this case, the safety "slow abort" and the "main circuit breaker" of a power converter). In this diagram, Q represents the unavailability, which corresponds to the PFD. In addition, Qm represents the mean unavailability, which corresponds to the PFD_{avg}).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

By using this tool, we have minimized the time of creating the models and performing both analysis. The calculations performed by Isograph can be exported in CSV (comma-separated values) format and integrated in our reports.

When the SFF data is not available, the Route 2_H from the IEC 61508-2:2010 Clause 7.4.4 can be applied. It is based on component reliability data from feedback from end users, increased confidence levels and hardware fault tolerance for specified safety integrity levels.

SIF Systematic Failures Analysis

Regarding the systematic failures analysis, the selection of the SIF components can be done by complying with the prior use approach from the IEC 61511-1:2016 Clause 11.5.3. In this case, it is necessary to gather sufficient evidence, showing that the dangerous systematic faults have been reduced to a sufficient low level compared to the required safety integrity. Another option is to apply the requirements for systematic safety integrity provided by the IEC 61508-2:2010 Clause 7.4 (Route 1_S, 2_S or 3_S).

SIF Application Program

The application program (AP), which contains the logic of all SIFs, is specified as part of the SRS and usually implemented in a PLC. The AP design and implementation requirements are defined in the IEC 61511-1:2016 Clause 12. In addition, the Annexes A and B from the IEC 61511-2:2016 provide guidelines and recommendations to produce an AP compliant with the standard. Most of our functional safety projects use Siemens Safety PLCs and, depending on the project, we work with Simatic Step7 or TIA Portal[1] programming environments.

[1] https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software.html

According to the IEC 61511-1:2016, the AP development cycle is divided in three phases: (1) design, (2) implementation and (3) verification.

The first phase consists of specification of the SIS logic and its operation modes in a clear and precise way, free from ambiguity and free from design faults. This functional specification is part of the SRS. The IEC 61511-2:2016 Annexes A, B, D and E show examples of recommended specification methods that can be applied to define the functional logic of a SIF. They also recommend the usage of model-based approaches, for example, in Annex B section B.2, the standard states: "The traditional text based approach of safety AP specification is not efficient enough to handle the advanced, complex safety requirements commonly found in SIF specifications. The most efficient tool to address these challenges is the Model-based design (MBD)...". In the Annex B section B.4.3.4 the standard states: "specification should be implemented in the graphical language of the model checking workbench environment...".

At CERN, depending on the nature of the project, we use one of these two popular model-based specification methods: Logic Diagrams (LD) or Cause and Effect Matrix (CEM). In [4], an example of how the CEM was applied to one of our projects is shown and it also introduces the tool that supports this formalism: SISpec. In addition, another prototype tool has been developed to support LD-based specifications for SIF logic: Grassedit. Both tools have the capabilities of test and verification cases generation for the verification phase.

In the implementation phase, the functional specification is translated into the PLC AP. Siemens safety PLC programs must be written in LADDER or FBD (Functional Block Diagram) with several restrictions in the data types, operations, etc. in order to be compliant with the requirements for LVL (Low Variability Language) defined by the IEC 61511. In addition, PLC brands normally oblige the PLC programmer to manually write the PLC safety program on their own pro-

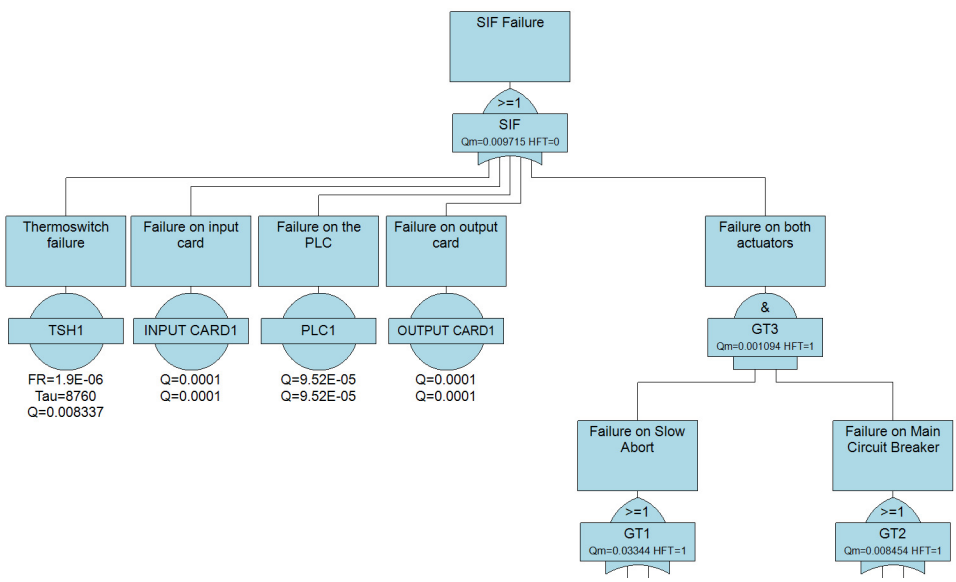


Figure 3: Fault Tree developed with the Isograph's Reliability Workbench.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

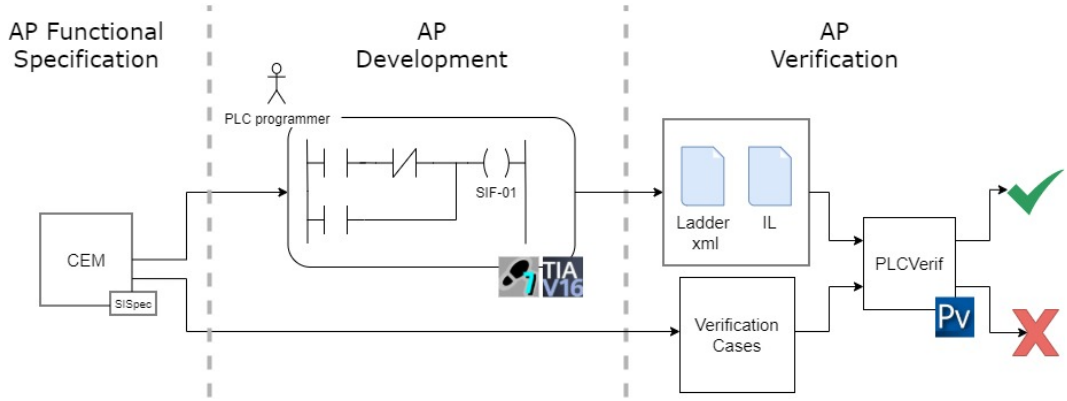


Figure 4: Specification, development and verification of the SIS AP.

gramming environment. This is the case for Simatic Step7 from Siemens. However, in their latest programming environment tool, TIA portal, it is possible to generate the AP source code from external tools, import it in TIA portal, compile it and produce a safety AP. This feature opens the door to automatic code generation of the PLC APs.

The verification phase consists of testing and reviewing the AP in order to confirm that the AP implementation corresponds to the AP design, and that there are no unintended states. In our case, in addition to the FAT (Factory Acceptance Test) activities (requirements defined in 61511-1:2016 Clause 13), we apply model checking to our Safety PLC programs, as recommended in Annex B (section B.4.3). By applying model checking, all combinations of the AP are checked to guarantee that the SIF logic from the specification (CEM for example) is respected in the implementation. For that purpose, we use the open-source tool PLCverif[2], developed at CERN [5]. An example of how PLCverif is applied to a safety PLC program can be found in [6].

Figure 4 illustrates these three phases and associates them with the software tools we use in the development.

SISpec, Grassedit, PLCverif and Isograph can contribute to improve the reliability of the SIS. They are also compliant with the IEC 61508-3:2010 Clause 7.4.4.2 "Software off-line support tools shall be selected as a coherent part of the software development activities". Therefore, it is certainly in the best interest of CERN to use these already available tools.

MANAGEMENT OF THE FUNCTIONAL SAFETY PROJECT

Management of functional safety projects is probably the most critical activity based on our experience. The requirements are described in IEC 61511-1 Clause 5. At CERN, typically the following roles are defined for each functional safety project:

- A functional safety expert: person with a deep knowledge and experience in applying the functional safety standards. The main activities of this role are the specification, design and validation of the SIS to be compliant with the risk reduction stated in the risk assessment. This person also participates in the risk analysis phase.
- A process expert: person with an excellent knowledge of the industrial process. The main activities of this role are to provide the operational requirements, contribute to the risk analysis and participate in the commissioning and validation of the SIS.
- The Departmental Safety Officer (DSO): person in charge of defining the tolerable risk levels and provide support to the project members in any safety matter. The main activities of this role are to conduct the risk analysis, define the tolerable risk level, as well as participate in the validation of the SIS.
- Health & Safety and Environmental Protection (HSE) unit representative: person that normally participates in the most safety-critical systems at CERN in coordination with the DSO and provides support in safety matters.
- Instrumentation and control experts: people in charge of selecting the hardware equipment for the SIS (sensors, final elements and controller), according to the SRS document, as well as implementing the AP.

In addition, for the Functional Safety Assessment (FSA), an external person to the project must be included in the FSA team. Moreover, an independent person to the project must conduct the Safety Audit.

Reporting

Documentation is an important part of the management of these projects. At CERN, report templates were created to cover different phases of the safety life-cycle. One of the most important and problematic aspects of the documentation is to keep the traceability between the documents of the project. For example, when a SIF specification is modified in the SRS, the proof testing document should be updated. We have created templates for the following reports: Hazard and risk assessment, SRS, FSA, proof testing and the safety manual. Traceability is currently maintained manually by the functional safety expert, but this is a topic taken into consideration for future work.

[2] https://gitlab.com/plcverif-oss/cern.plcverif

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 9: Tools, Methods and Report Templates for Our Functional Safety Projects

Safety life-cycle phase	Tools	Methods	Report templates
H&R assessment	-	FMEA and calibrated risk graph	Risk assessment report
SRS	SISpec and Grassedit	CEM and Logic Diagrams	SRS report
Design and engineering	Isograph, PLCverif and UNICOS (future work)	FTA, RBD, model checking and FAT	Design and verification report
Validation	-	-	Proof test
Management	-	-	FSA and safety manual

CONCLUSIONS

This paper shares our experiences in the design, development and management of functional safety projects that are based on the IEC 61511 standard.

The challenges and the adopted solutions for some of the most critical phases of the safety life-cycle have been discussed. Table 9 summarizes the current solutions adopted for our functional safety projects. In general, we have worked in two main topics:

1. The creation of report templates compliant with the standard. For example, the risk analysis and assessment, the SRS, the proof tests, etc. These templates help us to standardize our projects and speed up the creation of the necessary reports.

2. The integration of external tools that can contribute to improving the reliability of the final SIS and speed up the development, maintenance and verification time. This is the case of the CERN tools, PLCverif, SISpec and Grassedit and the commercial tool, Isograph.

Future Work

In general, we have faced the optimization challenge individually for each of the phases of the safety life-cycle. As mentioned previously in this paper, an unsolved challenge is to be able to keep traceability in an automatic way. We will explore the possibility of using commercial tools (e.g. exSILentia®) that tackle the problem of traceability and management of changes. We will also evaluate the possibility of developing our own tool that would be able to integrate all our templates, generate automatically an important part of our reports, integrate our existing tools and guarantee the traceability of the project.

In terms of management, we are currently working on the optimization of the workflow of our projects, defining more precisely the different roles and responsibilities of the different groups involved in a functional safety project.

Finally, we want to go a step further in the design and development of the SIS AP. At CERN, we have a large number of control and safety systems and we are always looking for the standardization of our PLC programs and supervision interfaces. For that purpose, we will aim to create a new component of the UNICOS[3] framework that is able to generate automatically the SIS AP from a formalized specification, respecting the IEC 61511 guidelines.

[3] https://unicos.web.cern.ch

REFERENCES

[1] "Functional safety - safety instrumented systems for the process industry sector," International Electrotechnical Commission, Geneva, CH, International Standard, 2016. https://webstore.iec.ch/publication/24241

[2] "Functional safety of electrical/electronic/programmable electronic safety-related systems," International Electrotechnical Commission, Geneva, CH, International Standard, 2010. https://webstore.iec.ch/publication/5515

[3] S. K. Hurst, H. Boukabache, and D. Perrin, "Overview of a Complete Hardware Safety Integrity Verification According to IEC 61508 for the CERN Next Generation of Radiation Monitoring Safety System," 5 p, Oct. 2020. http://cds.cern.ch/record/2771471

[4] B. F. Adiego *et al.*, "Cause-and-Effect Matrix Specifications for Safety Critical Systems at CERN," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 285–290, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-MOPHA041.

[5] E. B. Viñuela, D. Darvas, and V. Molnár, "PLCverif Re-engineered: An Open Platform for the Formal Analysis of PLC Programs," in *Proc. ICALEPCS'19*, (New York, NY, USA), ser. International Conference on Accelerator and Large Experimental Physics Control Systems, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 21–27, ISBN: 978-3-95450-209-7. DOI: 10.18429/JACoW-ICALEPCS2019-MOBPP01.

[6] B. Fernández Adiego, I. D. Lopez-Miguel, J.-C. Tournier, E. Blanco Vi nuela, T. Ladzinski, and F. Havart, "Applying model checking to highly-configurable safety critical software: the SPS-PPS PLC program," presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, Oct. 2021, paper WEPV042, this conference.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR03

THE FAST PROTECTION SYSTEM FOR CSNS ACCELERATOR

Y.L. Zhang†, P. Zhu, D.P. Jin
Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China

Abstract

The CSNS (China Spallation Neutron Source) consists of an 80 MeV H^- LINAC, a 1.6 GeV Rapid Cycling Synchrotron, two beam transport lines and one tungsten target and three beam lines. The designed proton beam power is 100 kW in Phase-I [1]. The fast protection system plays an important role to guarantee the safe operation of accelerator. The response time requirement for the CSNS fast protection system is less than 10 μs. The design of CSNS FPS was based on FPGA technology, and the VME bus and SFP connector was adopted for the hardware design. Different beam interlock and mitigation logic was designed so as to fulfil the operation requirements.

INTRODUCTION

CSNS accelerators are designed to accelerate very high intensity proton beam, Fig. 1 shows the schematic layout of the CSNS facility. The uncontrolled beam loss may permanently damage or give a high radiation dose to the accelerator components along the beam line [2, 3]. Therefore, high reliability for machine protection system is the basic requirement. The accelerator protection system must be carefully designed so that we can avoid the unnecessary beam loss. Besides, the availability, scalability, maintainability and the budget were also need to be carefully considered in the design and implementation stage.

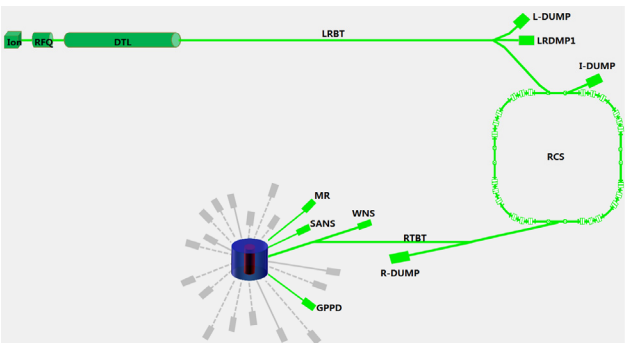


Figure 1: Schematic diagram of CSNS.

CSNS accelerator protection system consists of two protection systems, one is the PLC based slow protection system: NMPS (Normal Machine Protection System), the other is the FPGA based FPS (Fast Protection System). The response time within 20ms and 10μs are required for NMPS and FPS respectively [4]. The overall protection systems for CSNS as shown in Fig. 2. The output signals of PPS (Personnel Protection System) and TPS (Target Protection System) are input into the two NMPS main stations. RMS (Run Management System) can be seen as the beam permit system. Three beam stopping actuators are designed for protection system, which including ion source timing, post acceleration power supply of ion source and RFQ power source.

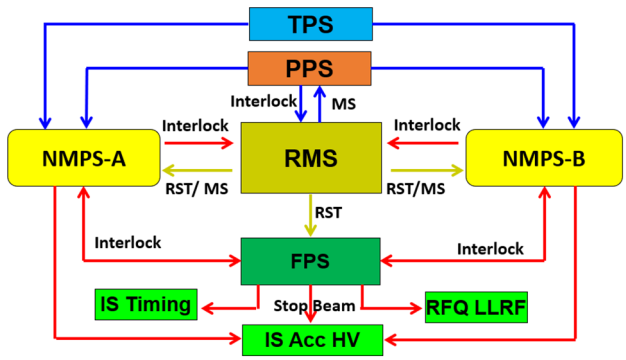


Figure 2: Relations among the protection systems for CSNS facility.

OVERVIEW OF THE CSNS FPS

The FPS for CSNS accelerator was designed based on the FPGA, the hardware was designed in VME form. In order to unify the hardware type, two kinds of boards were designed, one is the main logic board as shown in Fig. 3, the other is the optical signal input board as shown in Fig.4.

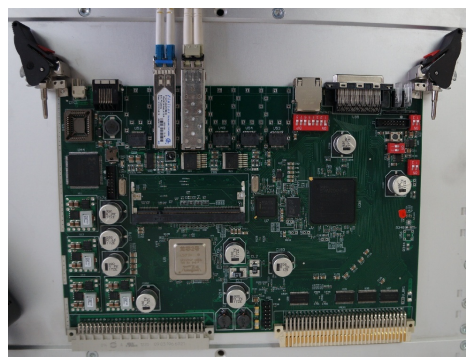


Figure 3: Picture of the FPS main logic board.

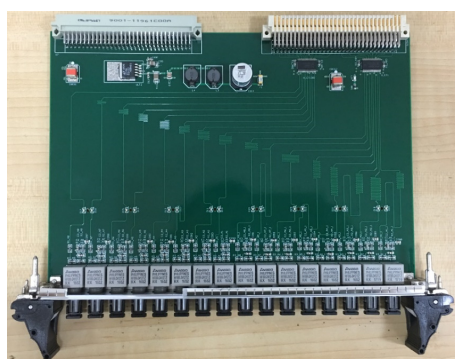


Figure 4: Picture of the FPS signal input board.

The high-speed serial communication technology was adopted by the main logic board, and the common plastic

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR03

optical connector was utilized by the signal input board. The main logic board can be programmed to perform different function, which can be used for signals convergence or the beam interlock.

The tree topology was adopted by the FPS. In the actual deployment, FPS consists of three layers, the signal input layer, the signal convergence layer and the beam interlock layer. Figure 5 shows the topology diagram of the FPS.

The FPS consists of one beam interlock station, two signal convergence stations and several signal input stations. The beam interlock station is responsible for carrying out the shutdown of beam or mitigation measures. The signal input stations are distributed near the corresponding devices. The total number of input signals is 304, and the signals are from DTL power supplies, beam loss monitors and Linac RF system.

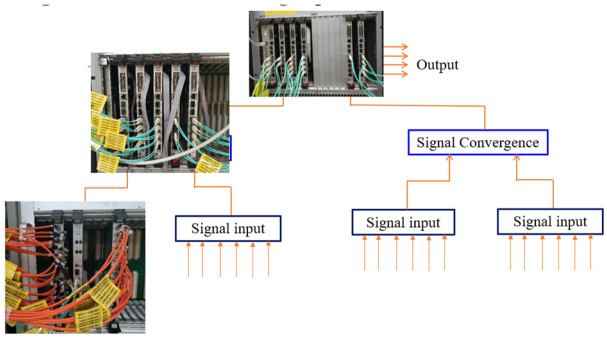


Figure 5: Diagram of the FPS topology.

BEAM INTERLOCK LOGIC

The input signals of FPS from different devices has different type. For example, the over threshold signal from beam loss monitor is in pulsed type, the fault signal from power supply will be in high level state. So, according to different input signal type, different action will be carried out by FPS. The following will introduce the detail of the interlock logic and actions.

Logic of Mitigation

For the pulse input signals, the mitigation measures will be carried out by FPS. When FPS receives pulsed signal, it will generate the output to inhibit next 1 second beam. Figure 6 depicts the output logic of inhibiting next 1 second beam. Figure 7 shows the details of mitigation actions, FPS will switch off the timing triggers to ion source and RFQ power source for 1 second, and after 1 second, the timing triggers will back to normal so that the beam will be recovered automatically.

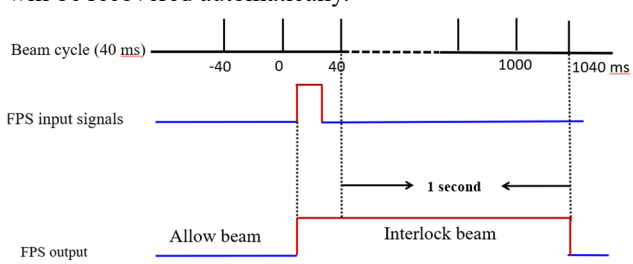


Figure 6: The logic of inhibiting the next 1 second beam.

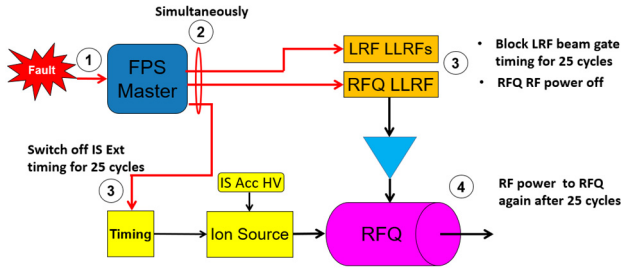


Figure 7: Actions of mitigation measure.

Logic of Shutdown of the Beam

Figures 8 and 9 show the logic of how the latched output is generated by FPS. When the FPS receives the fault signal, the mitigation measure will be taken place firstly. If the during time of fault signal is more than 5 seconds, or there are two pulsed inputs in 5 seconds, the output of FPS will be latched. The shutdown of the beam actions will be taken place.

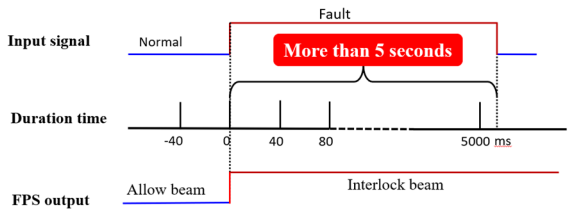


Figure 8: The latched output logic for PS input.

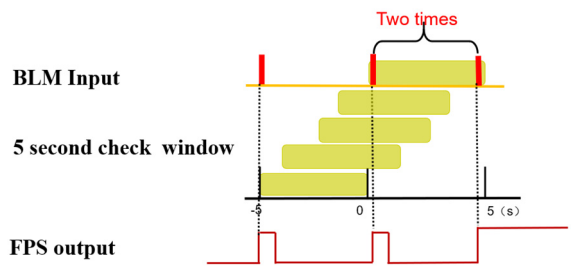


Figure 9: The latched output logic for BLM input.

Figure 10 show the detailed actions of shutdown of the beam. If the output of FPS changed to latched state, FPS will send interlock beam signals to MPS and ion source acceleration power supply, then the machine status will be changed to BEAMOFF and the ion source acceleration high voltage will be switched off. In this case, the manual beam recover process should be carried out in order to resume the beam.

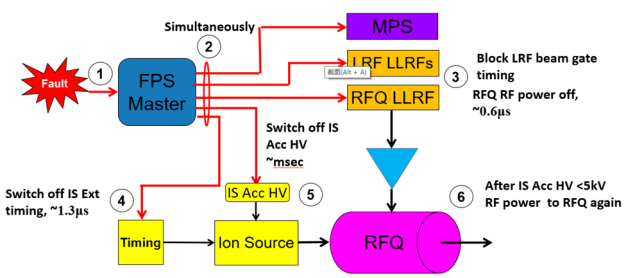


Figure 10: Actions of shutdown of the beam.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR03

CONCLUSION

The fast protection system for CSNS accelerator has been put into operation for more than three years. Different beam interlock logic was designed to fulfil the operation requirements. The fast protection system runs smoothly and reliable. The hardware will be upgraded in the CSNS-II project.

REFERENCES

[1] S. Wang *et al.*, "Introduction to the overall physics design of CSNS accelerators", *Chinese Phys.,* vol. 33, pp. 1-3, 2009. doi:10.1088/1674-1137/33/s2/001

[2] C. Sibley, "Machine Protection Strategies for High Power Accelerators", in *Proc. PAC'03*, Portland, OR, USA, May 2003, paper ROPB001, pp. 607-611. https://jacow.org//p03/PAPERS/ROPB001.PDF

[3] M. Ross, "Single Pulse Damage in Copper", in *Proc. LINAC'00*, Monterey, CA, USA, Sep. 2000, paper MOA006, pp. 47-49. https://jacow.org/l00/papers/MOA06.pdf

[4] Y. Zhang, "The accelerator control system of CSNS", *Radiat. Detect. Technol. Methods*, vol. 4, pp. 478–491, 2020. doi:10.1007/s41605-020-00203-y

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

SAFEGUARDING LARGE PARTICLE ACCELERATOR RESEARCH FACILITY- A MULTILAYER DISTRIBUTED CONTROL ARCHITECTURE

Feng Tao*, SLAC National Accelerator Laboratory, Menlo Park, USA

Abstract

Personnel Protection System (PPS) at SLAC is a global safety system responsible for protecting personnel from radiation hazards. The system's functional design shares similar concepts with machinery safeguarding, though the complexity of PPS is much higher due to its wide geographic distribution, large number of devices, and multiple sources of hazards. In this paper, we will first introduce the multilayer distributed control system architecture of SLAC's PPS, which serves three beam programs, e.g., LCLS, LCLS-II and FACET-II, that co-exist in the same 4km linear accelerator infrastructure. Composed of 50+ sets of redundant safety PLCs and 20+ access control PLCs, SLAC's PPS has five layers: beam program, beam switching and permit, zone access control, zone safety control and sensor/shutoff subsystems. With this architecture, safety functions often involve multiple controllers across several layers, make it a challenge on system analysis, verification, and testing. Therefore, in this paper, we will also discuss functional safety related issues for this type of complex systems.

OUTLINE OF THE PAPER

In this paper, the machinery safeguarding concepts and the introduction of SLAC's PPS are given first. Then different layers of SLAC and their functions are explained. For the representative E-Stop function, how each layer of PPS control contributes to the safety integrity is analyzed in Section 3. The impacts on system integrity for such a large distributed system are discussed in Section 4, with proposed solutions. At last, some remarks are given in the conclusion section.

MACHINERY SAFEGUARDING AND SLAC'S PPS

Machinery safety has been a matured and important field for safety-critical control system applications [1]. With the adoption of IEC 61508 [2] and the concept of functional safety, significant progress has been made in various application fields to formalize requirements on typical safety functions and their integrity levels. System designers should follow those standards and use those formalized requirements as a starting point. In machinery sector, there are two functional safety standards, e.g., IEC 62061 [3] and ISO 13849 [4], using different performance metrics Safety Integrity Level (SIL) and performance Level (PL) respectively. In Europe, ISO 13849 has been widely used as a type B1 standard that many type C specific machine safety standards referred to.

SLAC is a large research facility. It has a 2 miles long linear accelerator (Linac), which is being used to generate either high power electron beam, or extremely bright x-ray laser for scientific experiments. For this reason, the whole facility can be treated as a large "machine" producing electron/x-ray. So those best practice from machinery safety can be applied to SLAC's PPS as well.

There are some similarities and differences between conventional machinery safeguarding with SLAC's PPS.

Common practices include:

- Dual redundant circuitry for system reliability
- Operators' search procedure to secure the area
- Personnel trapped key interlock
- Wide usage of machinery safety certified components, from laser scanner, Emergency stop, trapped keys, circuit breaker, to safety PLCs.

On the other hand, as a large research facility, SLAC's PPS is much more complex than a conventional machinery safety system. The complexity comes from four factors:

- Wide geographical distribution
- Large numbers of field devices to monitor/control
- Multiple sources of hazards to interlock
- Interface to many other complex systems.

Those factors combined altogether pose a design challenge for PPS, which must be a distributed global safety system to meet those challenges.

SLAC's 2-mile long Linac was built in 1960s, and it is the longest linear accelerator in the world. Nowadays, this Linac are serving three different beam programs: LCLS completed in 2009, FACET-II completed in 2020, and the superconducting (SC) LCLS-II, which is under construction and will start operation in early 2021.

Figure 1 shows the locations of three beam programs, each taking up one third of Linac for beam acceleration:

- LCLS-II SC beam: Linac West (Sector 00- Sector 09)
- FACET-II: Linac Middle (Sector 10- Sector 20)
- LCLS-I Cu beam: Linac East (Sector 21- Sector 29)

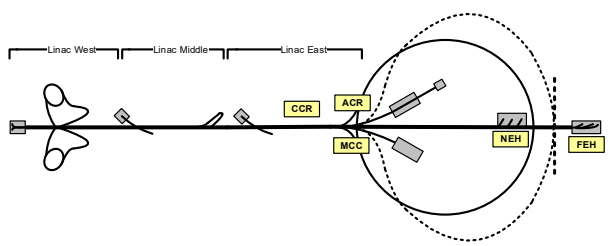


Figure 1: Layout of SLAC's Beam Lines.

* The author currently works for Underwriters Laboratories as a Functional Safety Staff Engineer,
† Feng.Tao@ul.com.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

The FACET-II beam will not pass Sector 20, but LCLS-I Cu beam and LCLS-II SC beam will enter Beam Switch Yard (BSY), where different routes and destinations of the beam are selected. Downstream to BSY is the Beam Transport Hall (BTH) and undulator complex, where electron beam wiggling through undulators to generate soft x-ray or hard x-ray. The x-ray laser travels across FEE-EBD to reach experiment hutches in Near Experiment Hall (NEH) and Far Experiment Hall (FEH).

To cover such a large facility, PPS needs to be distributed to effectively control/monitor field devices. Legacy systems use long haul trunk cables to connect field devices to control racks and control rooms, which is expensive and lack of flexibility. Just cite one example here: -48VDC was used in legacy systems to compensate voltage drop over a long distance. To modernize the system, we have used fiber-optic to create a dedicated ring topology network for all safety systems. Ethernet cables are been used as the infrastructure to minimize copper cable usage and to connect subsystems.

As each beam program has its own gun and acceleration RF devices, at each location, the PPS should be "aware" of not only the local prompt radiation hazards, but also hazards from all sources. Therefore, information exchange between different beam program is unavoidable.

At SLAC, attributes that distinguish PPS from other safety systems are strict configuration control and annual proof testing (bi-annual for some testing facilities onsite). Those are also the reasons that PPS are being trusted and being requested by many other systems (through interface signals) as a reliable means to bring the system into the safe state.

DIFFERENT LAYERS OF PPS

As a complex global safety system, PPS needs to meet the needs from different beam programs. It is a loosely connected system that has multiple layers and multiple installations. As a distributed system, it contains five layers:

- Global Beam Programs
- Beam Switching and Permit System (BSP)
- Zone Access Control
- Zone Safety Interlock Control
- Sensor/Shutoff Subsystems

Each layer includes functions that are important to maintain personnel safety, and they are all part of the hazard mitigation scheme. In this section, we will describe functionality of each layer in details.

Global Beam Program

There are three global PPS systems, corresponding to each beam program, and are conveniently named by their locations: e.g., Linac West Global PPS, Linac Middle Global PPS, and Linac East Global PPS. Each global PPS has separate top level safety controllers, to maintain the separation from other beam programs. This separation is especially important as each program have its own operation and maintenance schedule. Shutoff of one beam program should have no (or minimal) impacts to the operation of other beam programs.

Beam Switching and Permit System (BSP)

In the BSY region, there are a group of 5 sets safety controllers that are responsible for beam switching and stopper permit/control. This system determines if both hard x-ray and soft x-ray beamline configurations are correct, and issues permits for stoppers, magnets, kickers and septum in BSY. A detailed BSY beamline layout with stoppers configuration is shown in Figure 2.

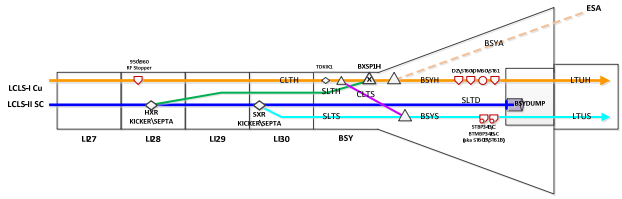


Figure 2: BSY Layout and Beam Stoppers.

As majority of stoppers in BSY cannot hold up to the powerful beam indefinitely, this system needs to work with upper layer control, to shut off beam before damages are made.

Zone Access Control

Each individual PPS zone has an access control system responsible for non-SIL rated protection functions, such as zone search procedure, trapped key release, magnetic door lock, area access control, audio/visual warning etc. Although those functions are not deemed safety critical, they do provide some level of protection, and will be given credit in Layer of Protection Analysis (LOPA) [5]. They are part of the engineering control that reduces risk and lowers the SIL required for safety functions inside safety PLCs.

Other functions contained in access control are state machine for access states, communication with EPICS, sensors checking and maintenance (such as BTM fill/vent routines). For cybersecurity reason, all safety controllers do not directly connect to EPICS, but use access controllers as the bridge by connecting to those controllers using fieldbus modules. This configuration is to provide an air gap for safety controllers to prevent potential cyber-attacks. Access control PLC is also responsible for driving field devices such as E-Stop LEDs and Emergency-Exit buzzers to make safety control portion compact.

Zone Safety Interlock Control

Zone safety interlock functions are contained in zone safety PLCs. At SLAC, PPS always uses dual redundant architecture, so there are identical Chain A and Chain B safety PLCs for each PPS zone. In *normal* mode, faulted input to Chain A will force both Chain A and Chain B controllers to turn off outputs; while in *test* mode, this cross-interlock between two chains is being turned off to facilitate testing, making it easier to identify faulty components on a single chain.

Sensor and Shutoff Subsystems

Traditionally those subsystems are customized built chasses with a soldered relay assembly inside. They have

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

been modernized by using safety PLCs to lower lifecycle cost and to increase reliability. Typical sensors used by PPS are Burn Through Monitor (BTM), Beam Shutoff Ion Chamber (BSOIC) and Residual Dose Monitor (RDM). Shutoff devices usually are relays or contactors to disconnect power supply to radiation generating devices, or solenoid valves to insert beam stoppers into beamline. Those devices are permitted by PPS logic at upper layer, and they also need to provide reliable status to the upper layer for escalated shutoff or setting access to the accelerator.

Historically, outputs of BTM and BSOIC chasses are connected to a 17mA current loop, worked as a logic "AND" gate to sum up all discrete subsystems' "OK/Fault" status within a given area. This current loop is called "Secure Loop" as it carries the information on whether a given area is secured to turn on hazards. For status of shutoff devices, similar current loops called "Set Entry" loops were built to sum up radiation generation devices' OFF status in a given area, to indicate if the area is safe to access.

During upgrade, some current loop chasses have been eliminated by directly wiring individual input of the loop to a safety controller, which functions the same but have more diagnostics and is more reliable.

There are no clearly established guidelines on how to define and divide the boundary of PPS subsystems. It involves many project decisions when the project was planned. Global PPS uses Siemens distributed safety PLC family of products, such as S7-315F and the newer S7-1515F. The same fail-safe CPUs are used in access control PLC systems, but I/O modules in those systems are simply not fail-safe. Allen-Bradley ControlLogix is also being used in access control PLC in some PPS zones. Those two layers of PPS use large-size PLCs suitable for distributed applications. On the other hand, zone safety and sensor/shutoff subsystems are usually compact and use Pilz PNOZMulti safety PLCs.

Functional Block Diagram of Linac West PPS

For better understanding of different layers of PPS, a functional block diagram of the Linac West is given in Figure 3:

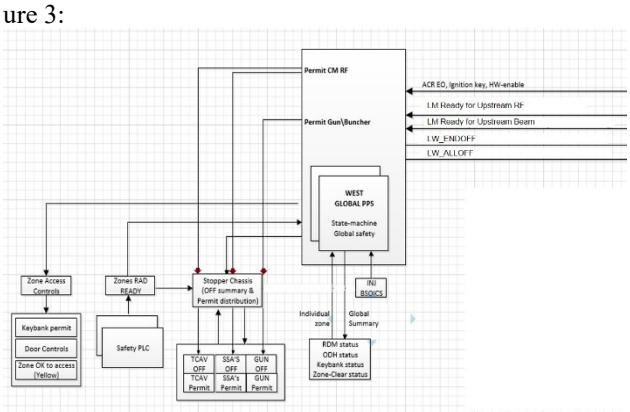


Figure 3: Linac West Functional Block Diagram.

Linac West is under construction for the LCLS-II project. The 10 sectors of Linac have been divided into three PPS zones: Inj-S00 for the injector, S01-S07 is the superconducting RF section, and S08-S10 will host normal conducting RF, as part of the LCLS-II-HE (High Energy) project scope, to boost electron beam energy from 4GeV to 8GeV. Each PPS zone has one dedicated access control PLC and two zone safety PLCs (Chain A/B), with the exception for S01-S07 zone. To effectively control such a large area over 700 meters, both access control and zone safety control use distributed architecture to make the field wiring manageable. To be specific, the zone access control using S7-1515F controller with 2 remote I/O drops, so that each I/O drop covers no more than 3 sectors. Zone safety control uses "SafeLink" communication modules to connect one "master" Pilz safety PLC with two "slave" PLCs, although the communication protocol itself is "peer-to-peer".

The West Global PLC is another distributed safety system at the top level to connect other distributed systems at lower levels. It will provide permits to devices that can generate radiation hazards when following conditions are met:

- Zone safety controllers in all 3 PPS zones indicate the entire Linac West is searched and secured
- Interface signal from Linac Middle indicates the downstream area (from Sector 10 down to BTH) is "READY" (searched and secured)
- Beam containment subsystems (BTM, BSOIC, RDM (Residual Dose Monitor)) indicate beam are contained and those subsystems are working properly.
- Interface signals from other safety systems such as ODH (Oxygen Deficiency Hazard), BCS (Beam Containment System) indicate no need for PPS to shut off the accelerator for them.

Linac West Global PPS issues permit signals to stopper control PLCs, which enable operators to turn on circuit breakers that provide power to 280 Solid State Amplifiers (SSAs) and traditional modulators (for LCLS-II-HE). Those SSAs are grouped into 20 facilities Interface Switches (IS) panels, using SIL rated 600A circuit breakers to shut off power supply to a group of SSAs reliably.

SAFETY INTEGRITY ANALYSIS

In a safety system design, the SIL or PL of a safety function is determined by risk assessment. For a standard machinery safety function, inputs, logic solver and final elements are all local devices. But this is not the case for large accelerator safety functions, making the analysis very tricky. In this section, we will use "Emergency Stop" function at different locations to demonstrate how different layers of PPS control work coordinately and affect the integrity of the overall function. Here the E-Stop function is chosen as the example because it is the most common safety function that required to achieve SIL 2 or PLd integrity performance by many safety standards.

Photon Experiment Area

For photon experiment hutches in NEH and FEH, PPS is relatively simple, and its settings are almost identical to those of standard machinery safety. Each experiment hutch is an isolated room with dedicated stopper for that beamline. In this system, safety inputs such as E-Stop, micro-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

switches installed on the perimeter boundary, are interlocked to the beam stopper using safety PLCs. The overall system is dual-redundant as well, with diagnostic test pulses applied to PLC's inputs whenever possible.

The E-Stop function in this area is affected by three PPS layers. The bottom layer is the hutch safety controller, which handles safety interlock logic; and the layer above is the access control PLC, which provides additional protection layers related to personnel safety. This helps to reduce the SIL required for those safety functions in the zone safety controllers [5]. For example, the perimeter door magnetic lock function will reduce the probability of challenging door microswitch interlock from outside of the hutch. The audio/visual warning function, which alerts people to leave the area also reduces the need of people left behind pressing E-Stop to initiate emergency shutdown.

In addition to those controllers, there is a diagnostic function implemented by upstream *PPS beam switching and permit system (BSP)*, which will insert the upstream beam stoppers in case the photon beam stopper fails to move in for any reason. This is achieved via a 17mA current loop named "LCLS Secure Loop", which passes through each photon beam stopper chassis in NEH. If any beam stopper has no permit and is not in "IN" position, relay logic inside the chassis will break the current loop. The current loop receiver will repeat the status and feed into BSP controllers to insert upstream stopper(s) in BSY. This function shall be treated as a diagnostic function applied to the output subsystem, photon stopper in this case, to provide a secondary shutoff path in case of the primary path failure. The safety function's reliability block diagram (RBD) is shown in Figure 4. As the overall system is dual-redundant, each block does contain symmetric components on two parallel paths.

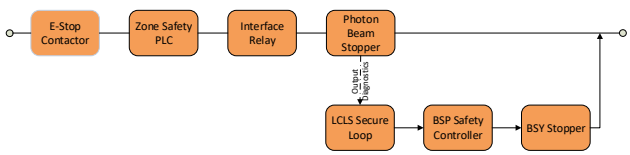


Figure 4: RBD for E-Stop Function.

In a previous paper by the author [6], the similar case has been studied, and the PFD value was calculated. Readers can compare the difference between two diagrams. In Figure 4, due to the presence of the additional diagnostic shutoff path, the safety function has a higher safety integrity. It is noticed that add-on diagnostics is a common strategy used in system design to improve system integrity. Such as using a Watch Dog Timer (WDT) as a secondary shutoff path, which can effectively upgrade the implementation from Category 1 to Category 2 for a machinery safety function [4]. But as on the primary shutoff path, dual redundancy should already provide sufficient reliability from electronics perspective; adding a secondary path may only be justified if the stopper has a substantial percentage of non-electrical failures from its failure mode analysis.

PPS in BTH and FEE-EBD

These sections are upstream to photon experiment area but downstream to BSY. There are no dedicated beam stoppers, so if any input to the zone safety controller faulted, the controller will de-energize its output to directly break the "LCLS Secure Loop". As stated before, this will cause the BSP to remove the BSY beam stopper permit, causing the stopper automatically move in. Therefore, in this case, zone PPS safety controller and BSP safety controller combined to be the logic solver during the safety integrity verification. The reliability block diagram in this case is shown in Figure 5 below:

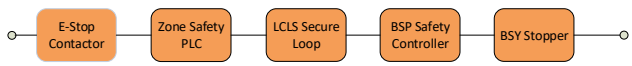


Figure 5: RBD for E-Stop Function at FEE-EBD and BTH.

In the figure above, the logic solver portion includes two or more safety PLCs (zone safety and BSP safety), LCLS Secure loop transmitter/receiver. Each Pilz safety PLC will add up about 150ms into the function's response time, so if the shutoff action needs to propagate further upstream to the Linac area and involves multiple subsystems, the overall response time should be verified against requirements.

Comparing Figure 4 and Figure 5, for the same E-Stop function, due to the different system configuration, the reliability performance differs, and the E-Stop function in photon area is more reliable.

PPS in Linac and BSY

The upstream LCLS-II SC beam has a bypass beamline all the way from Sector 10 to Sector 29 and have almost the same destination as LCLS-I beam. Therefore, once it is turned on, its radiation effects will affect downstream all the way to BTH. FACET-II does not have bypass line, but its radiation is dangerous to people in the LCLS-I Linac accelerator tunnel as well.

The impacts of such a layout are two folders:

- The safety functions triggered in a downstream area also include turning off upstream radiation sources.
- To turn on radiation generating devices in the upstream, all downstream areas need to provide enable signals to indicate it is safe to do so.

In both cases, the block diagram of the safety function will become more complex than usual, as more devices are showing up on the shutoff path, makes it even harder to meet the reliability requirements.

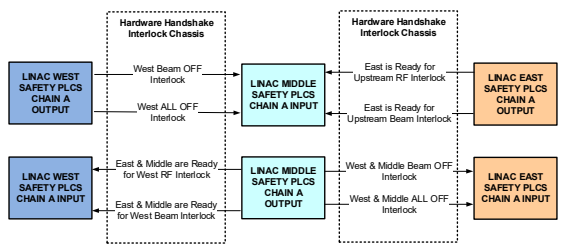


Figure 6: Interface Signals between Beam Programs.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

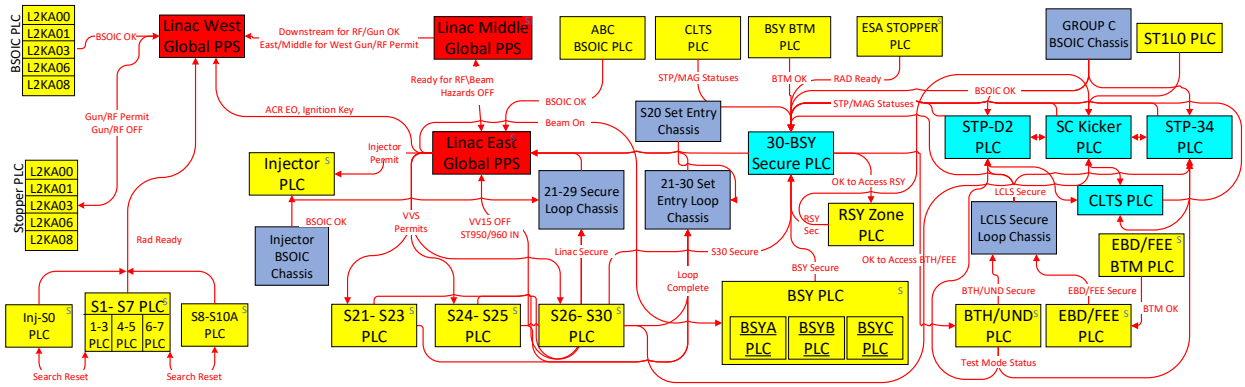


Figure 7: Signal Interfaces among PPS Safety Controllers and Safety Critical Electronics.

Figure 6 shows the interface signals among three global PPS systems. It should be noted that a beam program may have different modes to turn on beam, or RF or both. The hazard level is different for each operation mode.

As there are over 50 sets of safety controllers and safety critical electronic chasses across SLAC. There is a need to identify how those safety-critical signals flow through those subsystems for a better understanding. For SLAC's PPS at electron areas, a completed signal interface diagram is shown in Figure 7, where a letter "S" on the top-right corner of the PLC indicates it has a supervisory (access control) PLC.

This diagram is useful when to determine the safety shutoff action sequence, e.g., how the safety interlock broadcasts from one controller to another.

This diagram is also vital in interface control and test planning. For such a large system, PPS's annual site acceptance tests (SAT) need to be carried out by area and in multiple stages. Every interface signal in the diagram needs to be verified seamlessly on both ends to ensure the overall system functionality is intact.

Reading from Figure 7, we can identify the most challenging scenario for PPS safety interlock, e.g., E-Stop activation in BSY region. In this area, zone safety control also adopts the "master-slave" configuration. So if any E-Stop button wired to a "slave" safety controller is pressed, the shutoff actions will take the following sequence:

1. BSY zone safety "slave" PLC
2. BSY zone safety "master" PLC
3. 30-BSY Secure PLC
4. Linac East Global PPS
5. Linac Middle Global PPS
6. Linac West Global PPS
7. 5 Stopper PLCs (L2KA00/01/03/06/08)

For this scenario, there are 11 safety PLCs shown up in the reliability block diagram. Since all those PLCs are serially connected on the RBD, each PLC's PFH (Probability of Failure per Hour) value will add up to the PFH of the overall function.

If more than one beam program is running, then there are more controllers get involved in the shutoff action. For example, assume that both LCLS-I and LCLS-II beams are running and need to be shut off, then Linac East Global PLC also needs to remove permits sent to four zone safety controllers:

- Injector PLC (at Sector 20)
- S21- S23 PLC
- S24- S25 PLC
- S26- S30 PLC

In this extreme case, there are 15 safety controllers in total must function correctly to shut off all radiation hazards.

The following table from ISO 13849 [4] (Table 11) shows the relationship between number of subsystems, lowest PL and the overall system PL. Though the calculation is based on the average reliability value of each PL, but the message is clear: reliability lowers when more subsystems are connected in a simplex configuration.

Table 1: Calculation of PL for Series Alignment of SRP/CS

PL_{low}	N_{low}	➡	PL
a	>3	➡	None, not allowed
	≤3		a
b	>2	➡	a
	≤2		b
c	>2	➡	b
	≤2		c
d	>3	➡	c
	≤3		d
e	>3	➡	d
	≤3		e

Due to the large number of subsystems involved, in our case, it would be a challenge, if not impossible, for the E-Stop function to achieve SIL 2 integrity level!

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR04

Solutions for Meeting Integrity Requirements

There are several possible solutions to improve PPS's integrity:

- Add shielding to the boundary between Sector 9 and Sector 10 to eliminate the need to shut off upstream radiation source, thus simplifying the safety function.
- Re-define the shutoff function, make a clear distinction between interlock to gun and RF. Shutting off gun is more effective than shutting off RF, as RF alone only generates dark current, which is less dangerous than the accelerated beam.
- Simplify system architecture, reduce number of the PLC that re-routes the shutoff command. Safety relays and hardwiring may improve the safety integrity than safety PLCs.
- Add additional shutoff path.

For each option, there are some corresponding concerns as well:

- Adding additional shielding is expensive for the project, though this is the safest approach, and is consistent with the "inherently safe" principle.
- Before re-define the function, detailed study needs to be performed to narrow down the scope of shutoff, reduce the number of safety controllers on the shutoff path.
- Replacing PLCs with relays and cables is a reversal of the trend for more functional integration, modularity, and configurability. This will make the system less flexible and difficult to upgrade later.
- A good candidate for the shutoff path is to use Beam Containment System (BCS), which is another global safety system at SLAC [6]. This system is an independent system using different mechanisms to shut off the beam. Currently it is only being requested by PPS for a fast shutoff when BSY beam stoppers are in motion, so as to avoid the damage caused by beam hitting sides of stoppers. Using BCS as the secondary shutoff path needs additional work to connect two systems, but as there is no common cause factor between primary/secondary shutoff, the PPS's safety integrity can definitely meet the SIL 2 or PLd performance requirement.

Reviewing all options above, the best solution is to use BCS as the secondary shutoff path. The tie-in point from PPS to BCS should be close to the BSY, to maximize the performance.

CONCLUSION

SLAC's PPS employs a complex multilayer distributed architecture to protect personnel from prompt radiation hazards. This architecture is far more complex than the standard ones from ISO 13849 and IEC 62061 standards. In this architecture, there are more modular control systems deployed at different levels to meet the operation needs. However, this complexity inevitably lowers the overall system's integrity, which need to be considered and verified during the system implementation. For a successful safety system design, both functional requirements and integrity requirements must be met.

ACKNOWLEDGEMENTS

I would like to thank my colleagues in the Safety System Department: Enzo Carrone, James Murphy and Keith Belt. I deeply appreciate their supports during my ten years of adventure at SLAC! Accelerator Safety Officer Paul Miller and Michael Stanek, accelerator physicist John Schmerge and Jerry Yocky provided valuable information during discussions and design reviews.

REFERENCES

[1] D. Macdonald, "Practical Machinery Safety", Burlington, MA, USA: Newnes, 2004.

[2] IEC 61508, "Functional Safety of Electrical /Electronic /Programmable Electronic Safety-related Systems", 2nd Ed, IEC, 2010.

[3] IEC 62061, "Safety of Machinery- Functional Safety of Safety-related Electrical, Electronic and Programmable Electronic Control Systems", IEC, 2005.

[4] ISO 13849, "Safety of Machinery – Safety-related Parts of Control Systems", ISO, 2006.

[5] F. Tao, J. Murphy, "Applying Layer of Protection Analysis (LOPA) to Accelerator Safety Systems Design," in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, paper THCPA03, pp. 1217-1220, doi:10.18429/JACoW-ICALEPCS2017-THCPA03

[6] F. Tao, E. Carrone, J. Murphy, and K. Turner "Safety Integrity Level (SIL) Verification for SLAC Radiation Safety Systems," in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, paper TUC3O07, pp. 561-564, doi:10.18429/JACoW-ICALEPCS2015-TUC3O07

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR05

INTEGRATED SUPERVISION FOR CONVENTIONAL AND MACHINE-PROTECTION CONFIGURATION PARAMETERS AT ITER

D. Karkinsky†, W. Van Herck, I. Prieto Diaz, J. Soni, A. Marqueta, ITER Organization, St. Paul lez Durance, France

Abstract

Configuration parameters for ITER's I&C systems are predominantly high-coupled due to the nature of the process under control. Subsequently, I&C re-configuration requires an integrated supervision approach that addresses coupling through abstraction, automation, scalability, changeability, robustness and re-usability. Moreover, high-coupling might manifest at any tier of the I&C, and certainly spans configuration parameters across both conventional and investment-protection I&C.

Stemming from ITER design guidelines, the handling of investment-protection configuration parameters needs to meet the goals of IEC61508. These goals are mostly in congruence with the main concerns of integrated supervision identified above. However they also extend requirements that bind the supervision process with traceability and audit capabilities from sources to final self-test (run-time) diagnostics.

This presentation describes the provisions for integrated supervision at ITER and elaborates how these provisions can be used to handle machine-protection parameters in compliance with IEC61508.

INTRODUCTION

The ITER plant configuration system is a component of the Control, Data Access and Communication (CODAC) Supervision and Automation (SUP) system and is tasked to:

- Derive machine parameters from the central planned experiment information contained in the pulse schedule,
- Conduct a multi-stage engineering verification process involving a wide range of codes (e.g. electromagnetic induced forces on mechanical structures, scarce resource budget management, etc.),
- Convert machine parameters to the number and format representation of the various Plant Systems Instrumentation and Control (I&C), and
- Eventually load machine parameters to the Plant Systems Instrumentation and Control (I&C) as part of the experiment preparation.

The ITER plant configuration system interfaces to ITER machine Operation, to a heterogeneous set of data repositories (e.g. pulse schedule queue, machine geometry and condition, operating limits, live measurements, Plant System self-description data, etc.), and the Plant System Instrumentation and Control (I&C) systems that compose the ITER machine.

† Damien.Karkinsky@iter.org

The ITER plant configuration participates to the ITER defence-in-depth machine protection scheme by ensuring the configuration is as thoroughly verified as deemed necessary before starting lengthy and costly operations.

The baseline design for the Plant System I&C configuration interface is using EPICS records databases and Channel Access (CA). This was challenged during the 2014 CODAC design review. It was then understood that this choice was ill adapted to address the complexity required in the scope of ITER Plant Systems, and in particular in those areas below:

- Large and complex data structures involved.
- Existence of dependencies between parameters.
- Exception handling (e.g. restoration of valid configuration after a failed verification by the Plant System) and reporting.
- Handling of investment protection parameters for the Integrated Control System (ICS).

As a result, the configuration system was designed to support:

- Structured configuration variables,
- Atomicity of loading such, possibly complex, data structures,
- Protection against data corruption.

The protocol for Plant System I&C configuration is defined to follow the sequence outlined in Figure 1. The hash provides protection against data corruption and acts as a digital signature of over a data stream that can encapsulate an arbitrary set of configuration parameters.

In this presentation we report the results from an investigation into how the ITER plant configuration system can integrate with ITERs integrated control system (ICS) for handling investment protection parameters. As per ITER guidelines the ICS needs to meet the goals of IEC61508[1]. We report on the process by which we arrived at an adequate integration point for both systems and the technological solutions that have been put in place in support of this integration.

SUP SYSTEM DESIGN

ITER defence-in-depth principles, and the overall complexity of the machine, dictate that parameters are verified before being loaded to the plant. Distinct verification processes may be used depending on the nature of the task.

Furthermore, high-level operation will define operation goals, from which Plant System parameters must be derived (e.g. required cryogenic cooling capacity derived from predicted thermal loads).

To accommodate these requirements, the SUP configuration framework contains the following subcomponents:

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR05

- Configuration Verification and Validation Framework (CVVF): this framework encapsulates user-defined transformation and verification codes and exposes these as network services; the framework allows for the integration and invocation of codes that were implemented in different programming languages (e.g. C++ or Python);

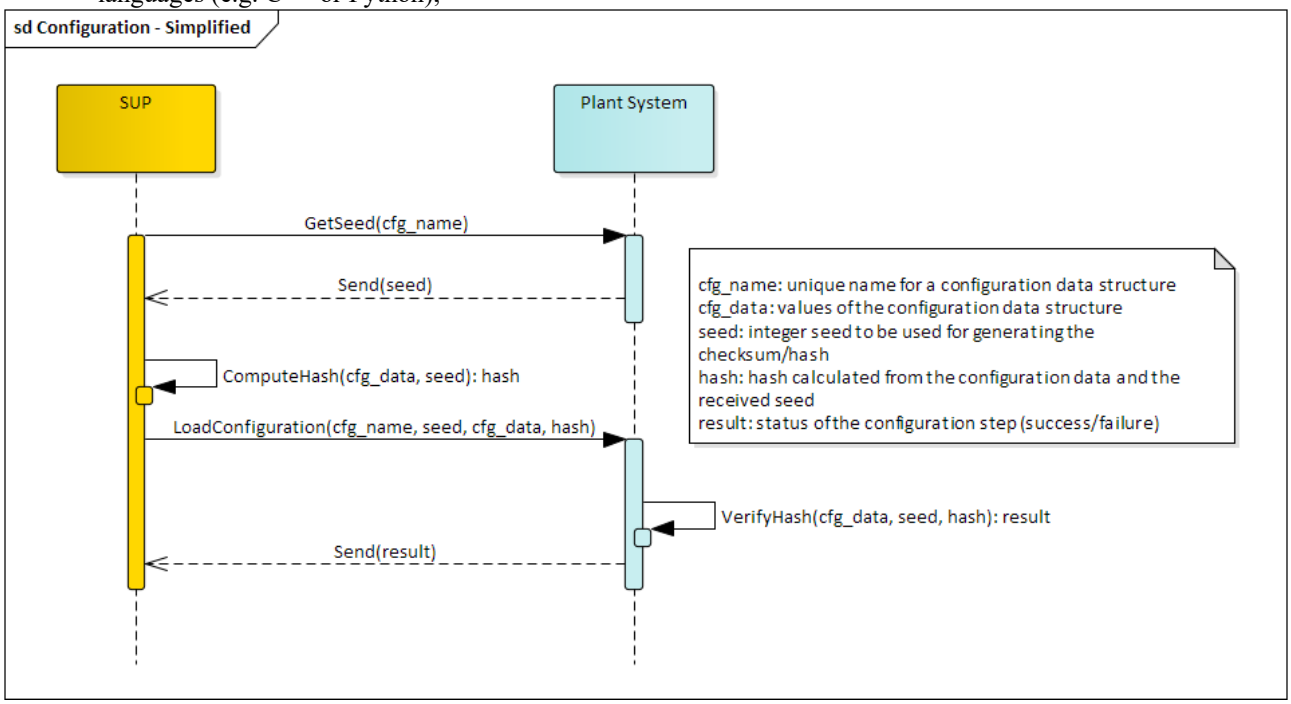


Figure 1: Simplified sequence diagram for Plant System configuration by SUP.

- Chain Data Processing engine (CDP) to define and execute workflows through CVVF applicable to various Plant I&C.

In order to support differentiated workflows for distinct types of users (e.g. operator and expert users), or during distinct life-cycle phases (e.g. testing, commissioning or operation), the configuration process and workflow definition is data driven.

An example of a simple configuration workflow clarifies how different processes in the workflow can depend on one another:

- Retrieve high-level parameters for the current activity;
- Call CVVF service to:
  - verify the authenticity of the received parameters;
  - verify that the high-level parameters of the current activity are within operational constraints;
  - transform high level parameters from the physics/operational domain to the engineering domain (machine specific);
  - verify that the machine specific configuration parameters do not lead to violations of machine or operational limits and constraints;
  - hash the streams of output to each Plant System;
- If all CVVF steps pass without an issue, transmit those parameters to the Plant System I&C in an atomic transaction.

Each of the CVVF processes express the configuration transformation and validation needs, through a set of remote function calls, which are identified using a Uniform Resource Identifier (URI). These function calls are fed with variables from the engine's workspace and result in updating other variables in the workspace.

INTEGRATED CONTROL SYSTEM AND ITS CONFIGURATION

ITER functions for investment protection are implemented in dedicated Plant System I&C called Plant Interlock Systems (PIS). PIS are coordinated by interlock functions of the Central Interlock System (CIS) where transversal investment protection risks are identified. Together these systems form ITER's Integrated Control System (ICS).

In the baseline design, the configuration of CIS is performed at the CIS desk through a custom High-Integrity Operator Commands (HIOC) [2] application layer protocol based on an OPC-UA interface. This interface is independent from SUP and the configuration of the PIS. Stemming from high-coupling, in almost all situations concerning PIS configuration, SUP configuration is the only realistic solution. This results in a gap of requirements for the ICS on handling critical configuration parameters. For instance, PIS modification without central coordination at the level

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

of ICS can lead to un-intended side-effects that can affect fault-tolerance claims and the ability of ITER to remain in a given operational state. Additionally, even in the case of some CIS functions it has become clear that certain parameters need to originate from the pulse-schedule, traverse a SUP workflow, and be communicated to the CIS desk with the necessary degree of assurance.

For this reason CIS needs to play a role during PIS modification, if only to ensure that the pre-conditions during PIS-modify actions are met. This justified a need to seek closer integration between HIOC and SUP and to develop a centralised policy for handling configuration parameters.

IEC61508 COMPLIANCE AND SUP

Investment protection function reliability implemented in ICS is commensurate to integrity targets in the SIL-1 to SIL-3 range according IEC61508. The goal of IEC61508 compliance for investment protection at ITER, is to demonstrate assurance within IO that these functions meet the identified reliability target. In order to drive the requirements for HIOC and SUP integration, we performed an analysis of IEC61508 requirements w.r.t. handling configuration parameters and matched these to SUP requirements.

From the perspective of investment protection, SUP generates outputs which can directly contribute to the executable code of a PIS and therefore, it was considered to be a T3 off-line support tool (see IEC61508-3 7.4.4) for the ICS. There are two main clauses of the standard that we need to respect with regards to such tools:

a) Treat SUP as a software element of the ICS where the level of reliance that is placed is assessed, failure mechanisms identified and mitigation measures taken (see IEC61508-3 7.4.4.1 and 7.4.4.5),
b) Define and meet criteria for coherency of this tool in the system development (which includes configuration) activities (see IEC61508-3 7.4.4.2).

In the case of a) only the workflows for storing, verifying and generating the configuration parameters for a PIS are of interest. Moreover, not all software components in a given workflow will need to be of integrity commensurate to the investment protection integrity target. Where parameters pass untrusted software components, adequate trusted measures can be added to close the vulnerabilities introduced from untrusted elements. In the context of PIS configuration, we aim to classify CVVF transformation and verification steps in three tiers:

1. Untrusted, where no guarantees can be placed on the step's correctness or it can introduce errors in the configuration parameters
2. Trusted with low-confidence, where the integrity of the step is not as required by the integrity target or there are intrinsic failure mechanisms in the technique on which the step is based, which cannot be mitigated.
3. Trusted with high-confidence – the integrity of the step is as strict as required by the integrity target and all failure mechanisms have been mitigated.

Confidence is directly related to the systematic capability (see IEC61508-4 3.5.9) with which a step was developed, verified and change managed. For instance; lack of separation of techniques, development teams or the presence of un-mitigated intrinsic vulnerabilities in the implemented method, reduce confidence in the step. However, low-confidence trusted steps can be combined into higher confidence following the constraints placed by systematic capability over elements (see IEC61508-2).

In the case of b) we set the criteria for structured configuration parameters from IEC61508-3 7.3.3.12-13, 7.4.2.14 and (guide) Annex-G. In a broad sense these require:

a. Verification of the consistency, completeness and compatibility of; data structures, operational parameters and interfaces.
b. Full transparency over the workflow and in particular identification of all items needed to replay any step in the process of loading configuration parameters.
c. Detection of unauthorised changes,
d. Detection of corruption at run-time.

Item a) is partially met from the design requirements of SUP. To meet the requirement fully it is necessary that PIS ROs and ITER Operators take formal steps in determining which configuration parameters pose investment protection risks (e.g. risk assessments) and to drive the specification for the workflows of trusted/untrusted steps within CDP from the identified risks.

In contrast, item b) was identified as a new requirement for SUP. This requirement is not immediately obvious from the perspective of conventional Plant I&C. The requirement requests SUP to ensure that inputs, step configuration, outputs and roles involved in executing each CVVF step at a point in time and in a given workflow, be recorded so that the chain can be audited from the source to destination. The ability to replay CDP workflows gives the ability to identify the source of a fault and drive a policy of continuous improvement.

Items c) is within the domain of ICS, where a central method is necessary to inhibit unauthorised changes, and item d) is in the domain of each PISs which needs to perform run-time verification over static-data as part of its self-check diagnostics.

Given these requirements, the interface point for HIOC and SUP was set at c) and d) and additional measures that will permit a configuration change to be traced to a workflow execution within SUP.

HIOC AND SUP INTEGRATION

The development of HIOC, followed IEC61508-2 requirements for a "black-channel" interface where measures necessary to ensure the failure performance of the communications process were implemented. IEC61784-3[3], which contains broad risks & measures guidelines for com-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR05

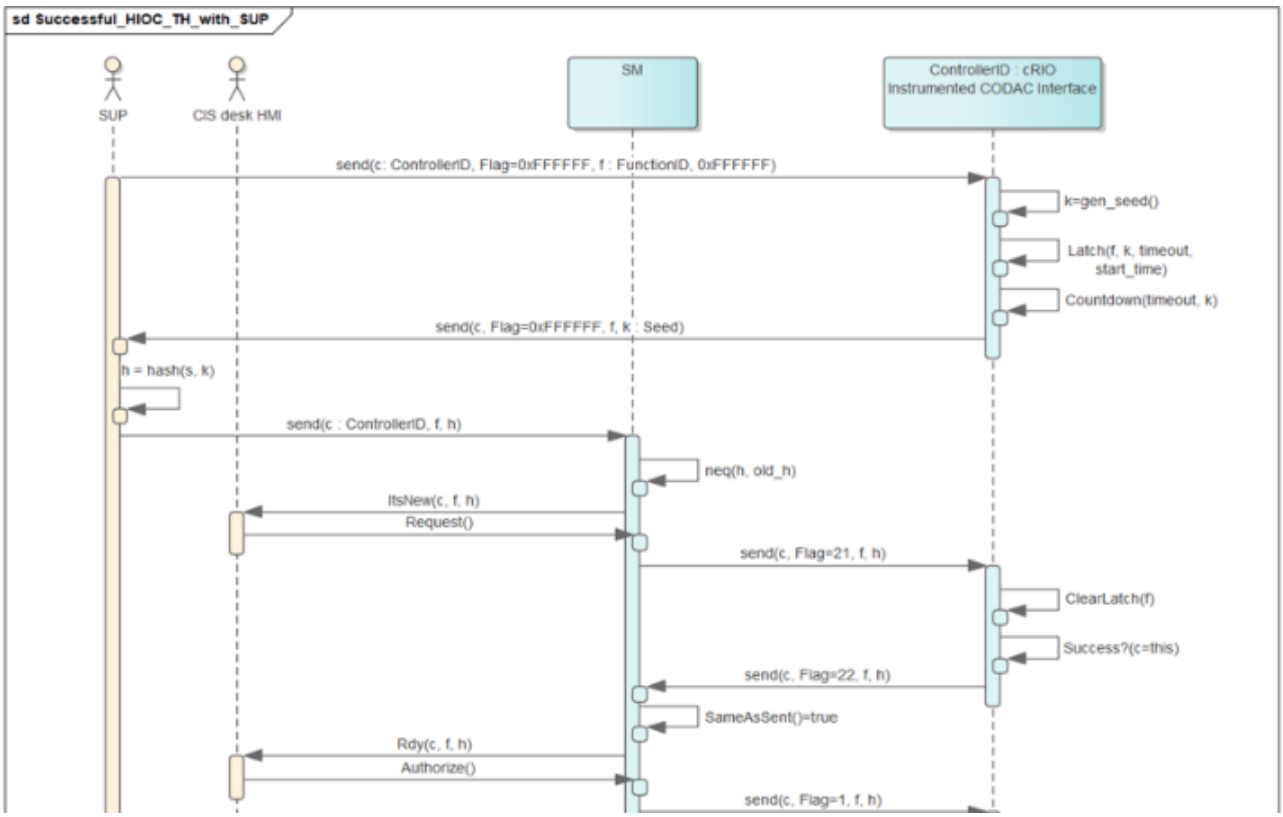


Figure 2: Interaction sequence in HIOC with SUP.

munication channels, was used as the source of requirements. The software elements that have been developed support up-to SIL-3 on PLC and SIL-2 on FPGA types of hardware.

HIOC involves a three-stage configuration process that aims to verify;

1. The controller identity (*ControllerID*) that is being modified is as intended by the operator.
2. The function identity (*FunctionID*) on the controller being modified is as intended by the operator.
3. The configuration parameter value (H) that is being changed is as intended by the operator.

These guarantees are provided at application level, meaning lower level components in the communications hierarchy can be of arbitrary integrity. If one of the three stages fails, HIOC aborts the configuration process which means that the parameter is not loaded into the critical path of the target function's control loop.

The main use-case of HIOC is to modify single value Boolean and Threshold parameters. Such parameters can be used to gate PIS special states e.g. mask the PIS response to a hard or a soft reset or set the maximum plasma current threshold for a given experiment.

In the case of HIOC with SUP integration, the Threshold use-case was re-purposed as an 8-byte SHA-1 hash of a configuration stream. Major assumptions in the use of HIOC is that it shall be utilised within a secure network where all entities are known and are non-malicious. It is within this assumption that HIOC provides a trusted mechanism to preload this hash and set the PIS's SUP interface in a state where the PIS can accept new parameters.

Each hash is valid for a specified time period and is based on a random seed value generated by the PIS. SUP requests the seed from the PIS on protocol initiation and generates the new hash (Figure 1). It then makes a request to the CIS Supervisor Module (SM in Figure 2). The CIS-SM initiates a HIOC interaction with the PIS to retrieve the seed and authorise the new hash. The first goal is to enter the hash, seed and time into the ICS logs. This permits the HIOC transaction to be traced to the outcome of the CDP workflow execution that produced the hash. The second goal of CIS-SM is to verify that given the current ICS state, the PIS can be modified. This can be done by asking for confirmation from the CIS desk (i.e. manually) or automatically. Rules on when PISs must be fully or partially available result from operator instructions. A fully automatic process is not possible as PIS operating rules vary.

Transmitting the hash through the remaining HIOC steps indicates an authorization to load the particular configuration stream that corresponds to the hash/seed pair.

On obtaining authorization to load new parameters, the PIS enters a wait-state for the parameter stream from SUP. During this state, the PIS might override the relevant investment protection functionalities (e.g. placing the outputs in a known disabled state to avoid a spurious trigger of ICS). If it receives this stream within the allocated timeout it can generate its own hash and compare the result against the hash received via HIOC. If the two hashes

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBR05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

match the PIS can proceed to load them into the critical control loop, which will then re-establish the investment protection outputs to their computed state. The result is reported to the CIS-SM in an abort or a success identifier which is also logged.

Additionally, the PIS can use the hash inside its self-test diagnostics loop to check if the parameters remain unmodified during runtime. Depending on the technology used to store these parameters, they could be susceptible to various forms of memory corruption (e.g. single event upsets).

This approach requires PISs to integrate with HIOC over a separate interface from SUP. The level of separation is dictated by the level of confidence that is required. At the extreme end a different hardware interface can be used. For instance, PIS typically integrate with SUP over the Plant Operations Network (PON). HIOC is normally executed over the Central Interlock Network (CIN), but it can be utilised on PON since the integrity is achieved at the application layer. Additionally EPICS and OPC-UA adaptors have been developed for the common frameworks used in PIS development [4].

CONCLUSION

In this presentation we reported on a central policy for handling investment protection configuration parameters of arbitrary complexity that arises from interfacing the ITER plant configuration system and the ITER integrated control system.

The requirements from this integration were derived from the IEC61508 standard as required by ITER guidelines. The presentation also outlined the technical design that supports this integration.

A number of open questions remain, such as the need of PIS developers to determine which parameters pose investment protection risks to drive workflow definition requirements and the need to incorporate runtime checks over the consistency of these parameters as part of self-test diagnostic routines.

The use of a central policy for handling configuration parameters and special state gate-keeping will eventually target the realisation of central rules over the level of readiness of components within ICS that govern the effective ITER Operational State or a proposed transition to a new state. In the present baseline, each PIS and CIS are preparing individual concepts of operation, and these plans are an important preliminary step towards identifying central rules for ITER operation.

REFERENCES

[1] IEC61508 “Functional safety of electrical/electronic/programmable electronic safety-related system”, 2010.

[2] Fernández Adiego, Borja & Blanco Viñuela, Enrique, (2017). “Applying model checking to critical PLC applications: An ITER case study”, in *Proc. ICALEPCS2017*, Barcelona, Spain, Oct. 2017, pp. 1792-1796. doi:10.18429/JACoW-ICALEPCS2017-THPHA161

[3] IEC61784-3 “Part 3: Functional safety fieldbuses - General rules and profile definitions”, 2010.

[4] A. Neto, F. Sartori, B. Bauvir, “Interfacing MARTe2 to the EPICS Channel Access and pvAccess protocols”, presented at EPICS collaboration meeting, Cadarache, France, Jun. 2019, unpublished.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

TEMPERATURE CONTROL FOR BEAMLINE PRECISION SYSTEMS OF SIRIUS/LNLS

J.L.N. Brito*, G. S. Baldon, F. R. Lena, M. A. L. Moraes, R. R. Geraldes, M. Saveri Silva, L. M. Volpe, Brazilian Synchrotron Light Laboratory (LNLS), CNPEM, Campinas, Brazil

Abstract

Precision beamline systems, such as monochromators and mirrors, as well as sample stages and sample holders, may require fine thermal management to meet performance targets. Regarding the optical elements, the main aspects of interest include substrate integrity, in case of high power loads and densities; wavefront preservation, due to thermal distortions of the optical surfaces; and beam stability, related to thermal drift. Concerning the sample, nanometer positioning control, for example, may be affected by thermal drifts and the power management of some electrical elements. This work presents the temperature control architecture developed in house for precision elements at the first beamlines of Sirius, the 4th-generation light source at the Brazilian Synchrotron Light Laboratory (LNLS). Taking some optical components as case studies, the predictive thermal-model-based approach, the system identification techniques, the controller design workflow and the implementation in hardware are described, as well as the temperature stability results.

INTRODUCTION

To address strict performance challenges of the new-generation beamlines at the Sirius 4th-generation light source [1], a series of innovative instruments have been developed in-house by the the Brazilian Synchrotron Light Laboratory (LNLS) over the past few years. Considered critical elements, special attention was given to optical systems – such as the High-Dynamic Double-Crystal Monochromator (HD-DCM) [2], the 4-bounce Crystal Monochromator (4CM) [3], mirror systems [4] and experimental stations [5].

These high-performance systems have been developed according to strong precision engineering and mechatronics principles, in which mechanical, thermal, metrology, control, and software aspects must be jointly addressed since early design stage [6]. Within this scope, sub-system functionalities are decoupled as much as possible from each other to minimized crossed effects, such that, as an example, a cooling system aims to be mechanically decoupled from the element of interest, whereas a positioning system/structure tends to be thermally decoupled from it.

Oftentimes, flexural structures are used as predictive solutions for kinematic mounts and fine positioning capabilities, with complementary thermal isolation possibilities [4, 7, 8]. Indeed, thermal decoupling is an essential aspect in many of these systems because cryogenic solutions were extensively adopted, either for sample conditioning [8], or for the silicon-based optical elements [2–4], to benefit from the thermal properties of this material at low temperatures in handling heat dissipation and thermal stresses/deformations.

Next, in addition to thermal decoupling, temperature control becomes an indispensable feature in many of these systems, being used for: 1) coarse static compensation in heat-flow for the desired operation temperatures; 2) heat load compensation (with and without beam, for example); 3) reduction of the time constants in the closed-loop systems as compared to passive or open-loop response; and 4) fine temperature control in drift-sensitive systems.

The following sections present the temperature control architecture that has been standardized for some Sirius beamline systems; the hardware, with appropriate actuators and sensors being critical for robust and high-performance control capabilities; predictive and experimentally derived plant models; and illustrative commissioning examples.

ARCHITECTURE AND HARDWARE

The hardware architecture adopted to control the temperature of the Sirius precision beamline systems is summarized in the Fig. 1, where a NI CompactRIO (cRIO) implements the control by reading the temperature sensors and acting through heaters present in the system, represented by a mirror. The cRIO also interfaces with the equipment protection system (EPS) [9], closing the beam shutter in case of overheating or disabling the cooling system to prevent overcooling, for example.

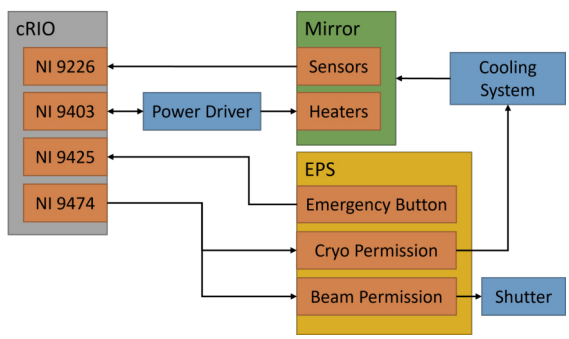


Figure 1: Temperature control hardware architecture.

The elements of the architecture and its characteristics are presented below.

Power Driver

The power driver is a multi-channel heater driver [10] in-house developed with 8 channels that delivers up to 1.5 A at 12 V DC per channel, where the voltage output

* joao.brito@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

is controlled via a 5 V PWM input signal. Each channel has also a 5 V TTL input to power enable and a 5 V TTL failure output to indicate that the power was tuned off due to an open-load or short-circuit.

cRIO

The cRIO platform has been adopted in solutions involving data acquisition, synchronization [11], temperature and high-dynamic motion control [12] for the beamlines, as a rugged platform that combines the high-speed logic and determinism of an FPGA with the flexibility of a real-time processor. The currently used cRIO chassis, i.e. cRIO-9039, cRIO-9045, cRIO-9048 and cRIO-9049, have 8 slots to which a variety of digital and analog I/O modules can be plugged in, allowing one or more solutions to be implemented on the same chassis.

In the temperature control setup, the EPS interface is done by one NI 9425, a 32-channel digital input module compatible with 24V logic level, and one NI 9474, a digital output module compatible with 5 V to 30 V. The chosen module for reading the temperature sensors was the NI 9226, an 8-channel RTD (Resistance Temperature Detector) analog input with 24 bits of resolution, 0 to 4 kΩ measurement range and a 3 and 4-wire connection compatibility. To drive the heaters, each channel in the power driver needs to receive two signals, namely, PWM and power enable, and returns a failure one. For this task the NI 9403 was selected, which is a 32- channel, 5 V TTL bidirectional digital I/O module.

EPS

The EPS is a safety application implemented in a programmable logic controller (PLC) that monitors diverse beamline systems to protect the instruments against incidents as vacuum loss, high temperature, power outage and more, as well as manages the interlock beamline system.

Cooling Systems

The general temperature control hardware architecture of most of the beamline instrumentation is based on a cryogenic heat sink, controlled thermal masses, and electric heaters as thermal actuators with sensing nodes determined by temperature sensors. For liquid nitrogen (LN_2) cooling systems like monochromators and cryogenic mirrors, that use closed-loop cryocoolers and open-cycle cryostats, respectively, the minimum heat sink temperature is determined by the vaporization temperature of the cooling liquid, being approximately 77 K at the current pressure configurations.

More recently, helium (He) pulse tubes started to be considered for forthcoming ultra-high vacuum (UHV) applications as well. As compared to the (LN_2) cryostats and cryocoolers, the He pulse tube prevents the consumption of the coolant, which is a significant operational advantage. Moreover, the cost of a typical beamline cryocooler is much higher than that of a typical cryostast or pulse-tube, which justifies its use only in very specific cases. In addition, the pulse tube should minimize operational issues that have been found in the cryostats, such as sensitivity to the phase transitions (icing, clogging and vibrations) and temperature oscillations in the load due to variations in the Dewar pressure. Indeed, in the pulse tube the flow is controlled by the frequency of the compressor, where are located the moving parts, that can be mechanically decoupled from the cold tip. Limitations in the pulse tube are essentially related to their load capacities, which are not sufficient for many beamline application.

UHV Heaters

The first cryogenic systems at Sirius beamlines utilized two different thin-film, UHV-compatible Kapton heaters from Taiwan KLC (part number TSC013D003GR36Z01), with nominal resistances of 36 Ω and 14.4 Ω for 4 W and 10 W power at 12 V, respectively. Although having an easy integration and proven vacuum compatibility, along with low cost, the flexible nature of the Kapton heaters made the clamping to the components a potential source of failure. Indeed, the components surface characteristics, together with human errors during the assembly process, often led to crushing of the electric track, or to bad contact interface, thus resulting in open or short circuits, overheating and failure. Also, it was found that these heaters might contaminate the vacuum systems in a failure event, due to outgassing. All these aspects reduce the systems robustness, increasing the need of corrective maintenance and posing risks to the sensitive optics.

Therefore, a new heating element is under development for higher reliability. As depicted in Fig. 2, it consists of an SMD (Surface Mount Device) nickel thin film and alumina power resistor from Susumu, soldered over a small aluminium metalcore PCB (Printed Circuit Board) using a lead free (SAC305) solder paste. The board is then encapsulated inside a small aluminium case using the Stycast 2850FT epoxy resin along with CAT11 catalyser. The aluminum PCB and housing serve as efficient heat condutors to the part of interest.

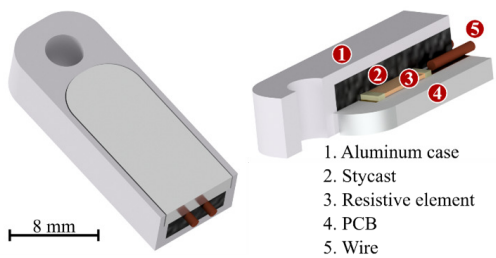


Figure 2: Heater 3D model and internal elements.

Heaters for 4.4 W and 8 W nominal power at 12 V were extensively tested for lifetime and outgassing by cycling them between 60 K and 390 K, as shown in Fig. 3. After 250 cycles, 3 of the 12 heaters tested failed, always within the first 20 cycles, which was observed by the heaters appearing as an open circuits for the power driver. For reference, initial tests with 7 heaters soldered with 63-37% tin-lead alloy did not present failures during all 82 tested cycles between 80 K

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

to 390 K, indicating that the lead-free solder could be the major culprit.

In fact, the lead-free soldering temperature is higher (220 °C versus 183 °C), making it harder to manually solder by hot air. Then, two possible solutions to the solder problem are: hiring the services of a specialized company for the soldering, allowing for a more consistent process; or ditching the lead-free solder altogether, since the tin-lead solder did not present those problems, which might offer residula risks in terms of vacuum compliance. One last aspect to consider is that, during the whole cycling test, no excessive outgassing was observed (see Fig. 3), which was an important step for validating this heater for UHV systems. Future monochromators, mirror systems and vacuum sample environments will be using the new heaters.

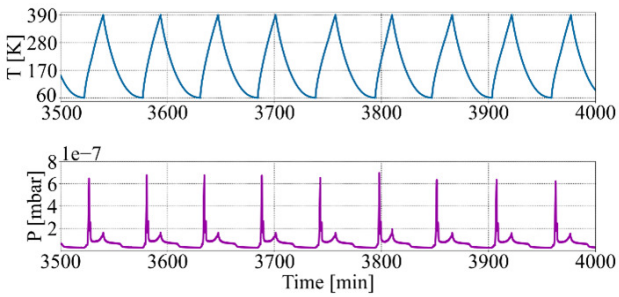


Figure 3: Fatigue and outgassing validation tests for the new heaters: cycling temperatures (top) and pressure (bottom).

Temperature Sensors

The temperature sensors also had design iteration since the beginning of the commissioning of the first cryogenic beamline instrumentation.Initially, 10k Ohm (0°C nominal) Platinum thin-film RTD sensors from IST (P10K.520.6W.B.010.D) were used for parts in operating temperature below 123 K, whereas ceramic Amphenol DC95F202WN negative temperature coefficient (NTC) sensors were used above 270 K, usually at room temperature components equal to 297 K. The part-though-hole (PTH) sensors were soldered to thin, 30 AWG, varnish insulated copper wires with small amounts of tin-lead (70/30) alloy. The set was then encapsulated with the same Stycast resin into small aluminium cases for thermal conductivity and mounting features.

The strategy with different sensors for different temperature intervals was chosen to better use the full sensing range of the NI 9226 (see Fig. 1), resulting in a temperature resolution better than 0.1 mK for both sensors at their nominal operational temperatures, as shown in Fig. 4. However, it has been recently concluded that, for most applications, it would be advantageous to trade the higher resolution by the possibility of measuring a broader temperature range, namely, the full range from the LN_2 cryogenic condition up to the optics typical baking temperature (> 85 °C).

Furthermore, the thin platinum wire of the 10 kΩ RTDs presented bad solderability and its assembly process was too laborious, resulting in unreliable mechanical bonds and a failure rate beyond acceptable for a robust beamline instrumentation. The alternative was to use 2 kΩ IST RTDs (P2K0.232.3FW.B.007) with custom-made flat gold-platted terminals, resulting in a full range sensor with better solderability and temperature resolution below 0.4 mK over the entire measurable range (see Fig. 4).

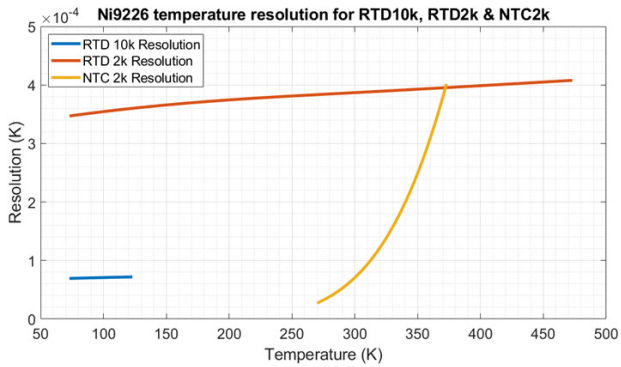


Figure 4: Temperature resolution over the measurable range for the three sensors and NI 9226 at high resolution mode.

The final design of the new sensor in depicted in Fig. 5. The encapsulating process is similar, with just slight geometry changes to simplify assembly. For validation, a few units were cycled for fatigue in the same test setup of the heaters (see Fig. 3), ultimately being totally operational. Dozens of these sensors are currently operational at the beamlines with no failure detected. Finally, as a next step towards improved control accuracy, a sensor calibration setup is currently in development and will be presented in a future work.

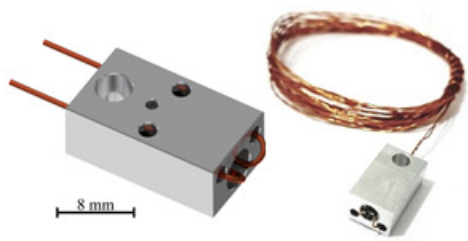


Figure 5: Temperature sensor assembly.

CONTROLLER DESIGN

Assuming that a certain knowledge about the plant to be controlled is necessary to specify the control architecture and its parameters, the controller design workflow described below initiates with predictive thermal modeling combined with a system identification method. After that, a model to estimate the plant behavior is defined and a proposed controller is simulated, implemented and validated with the real plant.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Predictive Thermal Modeling

To develop a predictive thermal model for each precision beamline system, lumped mass thermal models have been implemented together with auxiliary finite elements analyses (FEA) to simulate the system behavior under variable operational conditions, taking into account also non-linear aspects [4, 7, 13].

The predictive models have contributed to validate and improve thermal modeling, not only defining power requirements for control heaters and an initial guess to the controller parameters, but also diagnosing system issues, as the manufacturing limitations. Furthermore, during the design phase the lumped-mass models make the system parts sizing faster than by using only FEA. Recently, with experience and customized modeling tools, the differences found between model predictions and measurements have been below 10 K, as presented in [13].

System Identification

As mentioned, a relevant aspect about the beamline precision systems is that, although they are composed of different parts operating at different temperatures, these parts are designed to be thermally decoupled [4]. This has been confirmed during the predictive thermal model validation and also during the system identification process, in which the observed thermal coupling is negligible. Because of this, the systems can be seen as a set of isolated single-input single-output (SISO) plants.

Since the controller's goal is to keep the plant temperature stable, the system identification approach was used to find an approximate linear model of each SISO plant for a given operation condition defined by the desired operating temperature.

Taking the HD-DCM from the EMA beamline as a study case, which is a 14-SISO-plant system with 2 plants operating at −118 °C, 3 at −6 °C and 9 at 18 °C, system identification based on the step response was performed to estimate continuous-time transfer functions (TFs) for the plants, described in the Laplace domain by

$$\frac{Y(s)}{U(s)} = \frac{k}{(s-p)},$$

where s is the Laplace variable, $Y(s)$ is the sensor output in Celsius, $U(s)$ is the power input in W, k is the dc gain and p the pole of the transfer function.

After a set of data acquisitions with the system around the operating temperature for plant identification and TF estimation, another set at slightly different temperatures was selected for validation of the estimated plants. In the latter, the temperatures were taken only a few tens of degrees off, to prevent exceedingly large variations of the non-linear materials thermal properties. Figure 6 shows an example with results of both the identification and the validation sets, obtained for one of the HD-DCM crystals, the most critical element in that system whose desired operating temperature is −118 °C and maximum estimation error found during the validation process was less than 0.7 °C. For the other 14 plants, the estimation error remained below 1 °C.

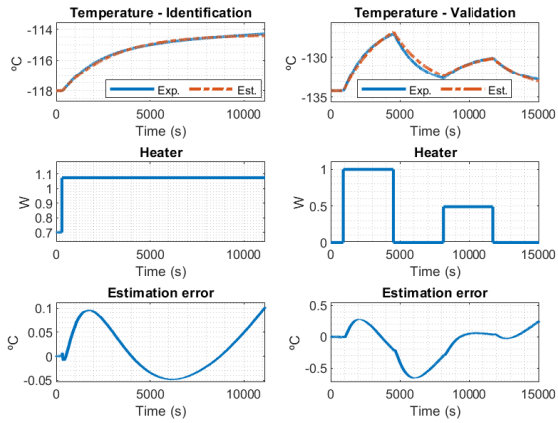


Figure 6: Experimental and identified estimation results of system identification (step response) (left) and validation (right) for the HD-DCM crystal at the EMA beamline.

Controller

Due to its robustness and ease of tuning and implementation, a proportional integral derivative (PID) controller was selected as the first choice to implement the temperature control for the beamline precision systems. As the controller should be implemented in a digital platform, it was designed in the discrete time domain, in which the ideal PID transfer function is represented by

$$PID(z) = \frac{U(z)}{E(z)} = k_p + \frac{k_i.T_s.z}{z-1} + \frac{k_d(z-1)}{T_s.z},$$

where $U(z)$ is the controller output, $E(z)$ is the controller input, defined by the difference between the temperature setpoint and the plant output temperature sensor, k_p is the proportional gain, k_i is the integral gain, k_d is the derivative gain and T_s is the sampling time.

However, the ideal PID has some undesired effects that need to be avoided. Firstly, about the derivative implementation, an instant variation in the temperature setpoint or even the noise produced by the temperature sensor can cause high derivative response, resulting in unreasonable control output. Thus, to prevent this behavior, the following PID with derivative filtering was implemented

$$PIDF(z) = \frac{U(z)}{E(z)} = k_p + \frac{k_i.T_s.z}{z-1} + \frac{k_d}{T_f + \frac{T_s.z}{z-1}},$$

where T_f is the derivative filter time constant.

Then, another undesired effect is the integral windup, caused by the continuous integration of the input $E(z)$ by the integral term even when the power driver output is saturated. Hence, an anti-windup scheme is implemented to turn off the integral term ($k_i = 0$) when the controller output reaches the limits of the power driver.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

Results

Once the plants were identified and the controller model was selected,the controller parameters were calculated to minimize the closed-loops settling time, while keeping the overshoot within less than 10% to a step response. Still considering the HD-DCM crystal from EMA, Fig. 7 shows the experimental and estimated closed-loop step response with a 0.05 °C variation in the setpoint input. In this case, the settling time to within 10% is only 10 min, as compared to the time constant of 1.5 h of the open-loop response, seen in the identification plot in Fig. 6. Similar performances were also achieved in the other controllers.

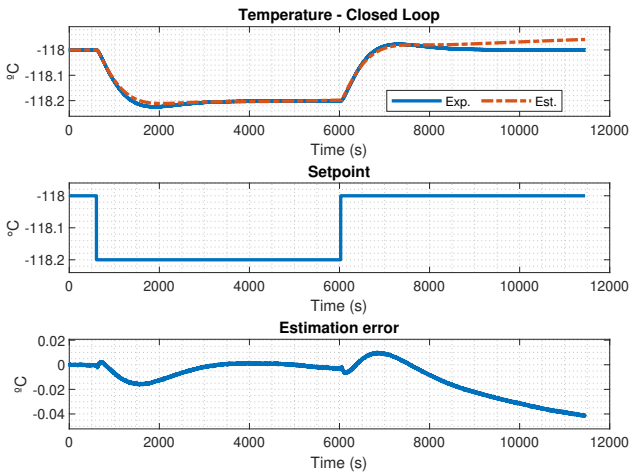


Figure 7: Simulated and experimental closed-loop step response for the HD-DCM crystal at the EMA beamline.

Then, looking at the long-term temperature stability, a variation of less than 0.004 °C is observed in the 12-hour plot presented in Fig. 8, with the larger dip possibly related to a refilling event in the cryocooler.

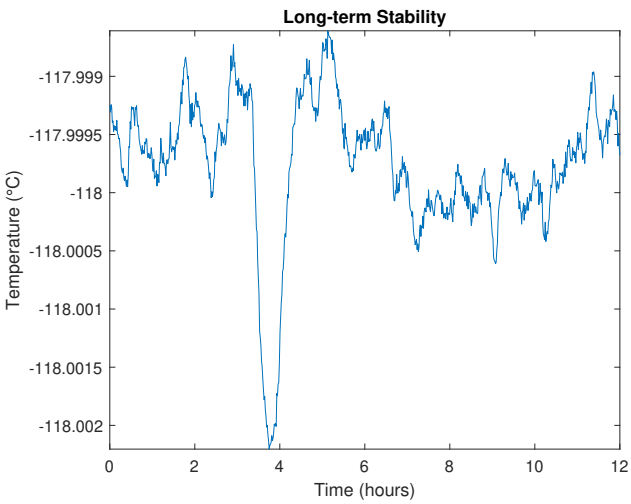


Figure 8: Long-term closed-loop thermal stability in the HD-DCM crystal at the EMA beamline.

CONCLUSION

This work presented the system architecture, the main hardware, and the controller design workflow adopted to the temperature control of Sirius beamline precision systems, where the predictive thermal modeling combined with the system identification approach contributed to define suitable controllers to achieve the design requirements. For reliable and robust beam delivery and experimental conditions, in addition to static compensation at nominal operating temperatures, control levels down to the mK range are, indeed, welcome and desired for ultimate performance in preserving wavefront and/or in positioning optical elements and samples. Over the commissioning time of the first Sirius beamlines models and procedures have been continually improved. The next steps are related to more systematically analyzing the influences of temperature stability on the photon beam position.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology and Innovation, and the contributions of the LNLS team.

REFERENCES

[1] Sirius project, https://www.lnls.cnpem.br/sirius-en/

[2] R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius", in *Proc. MEDSI'16*, (Barcelona, Spain), Jun. 2017.

[3] M. S. Silva *et al.*, "Four-Bounce Crystal Monochromators for the Sirius/LNLS Beamlines", in *Proc. MEDSI'20*, (Chicago, USA), May 2021.

[4] R. Geraldes *et al.*, "The Design of Exactly-Constrained X-Ray Mirror Systems for Sirius", in *Proc. MEDSI'18*, (Paris, France), Dec. 2018.

[5] R. Geraldes *et al.*, "Design and Commissioning of the TARUMÃ Sta-tion at the CARNAÚBA Beamline at Sirius/LNLS", in *Proc. MEDSI'20*, (Chicago, USA), May 2021.

[6] R. Munnig Schmidt, G. Schitter, A. Rankers, and J. van Eijk, *The design of high performance mechatronics: high-tech functionality by multidisciplinary system integration (2nd revised edition)*, English. Netherlands: IOS Press, 2014, NEO, ISBN: 978-1-61499-367-4. DOI: 10.3233/978-1-61499-368-1-i.

[7] M. S. Silva, R. Geraldes, A. Gilmour, T. Ruijl, and R. Schneider, "Thermal Management and Crystal Clamping Concepts for the New High-Dynamics DCM for Sirius", in *Proc. MEDSI'16*, (Barcelona, Spain), Jun. 2017, pp. 194–197.

[8] F. Lena, R. Geraldes, M. S. Silva, and G. Moreno, "A Cryogenic Sample Environment for the TA-RUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", in *Proc. MEDSI'20*, (Chicago, USA), May 2021.

[9] L. Arruda *et al.*, "Equipment and personal protection systems for the sirius beamlines", in *Proc. ICALEPCS'21*, (Shanghai, China), Oct. 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV001

[10] M. M. Donatti, D. H. C. Araujo, F. H. Cardoso, G. B. Z. L. Moreno, L. Sanfelici, and G. T. Semissatto, "Multi-channel heaters driver for sirius beamline optical devices", in *Proc. ICALEPCS'21*, (Shanghai, China), Oct. 2021.

[11] J. R. Piton, D. Alnajjar, D. H. C. Araujo, J. L. B. Neto, L. C. Guedes, and M. A. L. de Moraes, "Tatu: A flexible fpga-based trigger and timer unit created on compactrio for the first sirius beamlines", in *Proc. ICALEPCS'21*, (Shanghai, China), Oct. 2021.

[12] M. A. L. Moraes *et al.*, "The FPGA Control Implementation of the High-Dynamic Double-Crystal Monochromator at Sirius Light Source", in *ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems*, (Boston, USA), ser. ASPE 2020 Spring – Design and Control of Precision Mechatronic Systems, May 2020, pp. 131–136.

[13] L. M. Volpe, J. C. Corsaletti, B. A. Francisco, R. R. Geraldes, and M. S. Silva, "Thermal model validation for the cryogenic mirror systems for sirius/lnls", in *Proc. MEDSI'20*, (Chicago, USA), JACoW Publishing, Jul. 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

POSITION SCANNING SOLUTIONS AT THE TARUMÃ STATION AT THE CARNAÚBA BEAMLINE AT SIRIUS/LNLS

C. S. N. C. Bueno†, L. G. Capovilla, R. R. Geraldes, L. C. Guedes, G. N. Kontogiorgos, L. Martins dos Santos, M. A. L. Moraes, G. B. Z. L. Moreno, A. C. Piccino Neto, J. R. Piton, H. Tolentino, Brazilian Synchrotron Light Laboratory (LNLS), CNPEM, Campinas, Sao Paulo, Brazil.

Abstract

TARUMÃ is the sub-microprobe station of the CARNAÚBA beamline at Sirius/LNLS. Covering the range from 2.05 to 15keV, the probe consists of a fully-coherent monochromatic beam varying from 550 to 120nm with flux of up to 1e11ph/s/100mA after the achromatic focusing optics. Hence, positioning requirements span from nanometer-level errors for high-resolution experiments to fast continuous trajectories for high throughput, whereas a large flexibility is required for different sample setups and simultaneous multi-technique X-ray analyses, including tomography. To achieve this, the overall architecture of the station relies on a pragmatic sample position-ing solution, with a rotation stage with a range of 220°, coarse stages for sub-micrometer resolution in a range of 20mm in XYZ, and a fine piezo stage for nanometer resolution in a range of 0.3mm in XYZ. Typical scans consist of continuous raster 2D trajectories perpendicularly to the beam, over ranges that vary from tens to hundreds of micrometers, with acquisition times in range of milliseconds. Positioning based on 4th-order trajectories and feedforward, triggering including the multiple detectors and data storage are addressed.

INTRODUCTION

TARUMÃ [1] is the sub-microprobe station of the CARNAÚBA beamline [2] at Sirius/LNLS [3]. The CARNAÚBA beamline was designed to cover the range from 2.05 to 15keV, and is based on an all-achromatic optical design. The probe consists of a fully-coherent monochromatic beam varying from 550 to 120nm, focused by a set of KB (Kirkpatrick-Baez) mirrors, and with flux of up to 1e11ph/s/100mA. TARUMÃ was designed to perform simultaneous multi-technique analyses under a variety of conditions, including *in situ*, *in vivo*, and *in operando* experiments. Thus, it complies a number of customized sample environments [1, 4, 5] and a set of detectors of different types, including: two SDD Vortexes (Xspress 3 and Xspress 3X), two area detectors (MobiPix 45D and PiMega 135D, one XEOL detector, and an in-house spectrometer. Figure 1 shows a detailed view of the sample and the detectors (left) and an overview picture of the TARUMÃ station (right).

The sample stage is composed, from bottom up, of: an Aerotech's 300DL XY, a Newport's IDL280 Z20 wedge-type vertical stage, an Aerotech's ABRS 250MP air-bearing rotary stage, and a PI's P-563.3 XYZ stage. The planar and vertical stages are responsible for the rough alignment of the sample with a range of ±10mm in XYZ. The rotary stage, constrained over 220° due to cable management, is used for tomography, diffraction, and Bragg CDI experiments. A cable carrier system allows additional instrumentation, like special gases, heaters, sensors, and vacuum lines, to be used in the sample environment. The XYZ piezo stage is responsible for the sample scans with nanometer resolution over a 300µm range and has a payload capacity of 5 kg.

Resolutions down to the order of 10nm are desired from the experiment's reconstructions, which sets an upper boundary for the relative positioning errors between the sample and the KB mirrors. Thus, to reach these positioning levels, the KB optics and the sample stage are placed

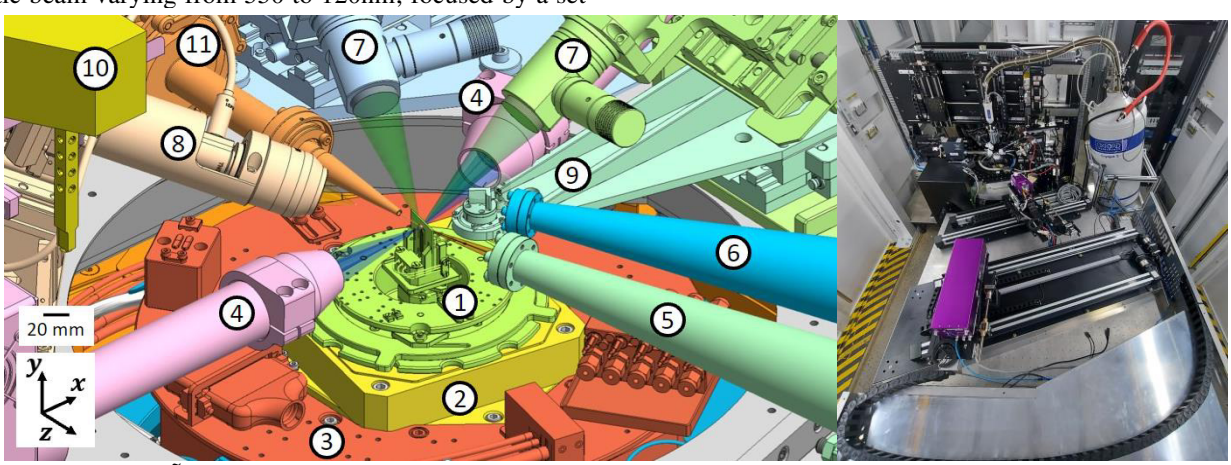


Figure 1: TARUMÃ station: drawing of the region surrounding the sample, detailing: the sample setup(1), the sample stage (2 and 3), the fluorescence detectors (4), the flying paths for the transmission(5) and the diffraction(6)area detectors, the optical microscopes (7), the XEOL optics (8), a crystal analyser spectrometer (9), the pick-and-place gripper (10), and the KB vessel exit port (11). (left) and assembly status on 10/07/2021 (right).

† cassiano.bueno@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

above a common granite bench [1]. Furthermore, to have a more accurate estimate of the sample position with respect to the focusing optics, a metrology target on the rotor of the rotary stage is measured by 3 capacitive sensors on an external metroframe fixed to the granite bench. Hence, the information of the capacitive metrology can be combined with the data from P-563.3 capacitive sensors to improve the post-processing reconstructions. Finally, the controller of each stage is designed and optimized for the system to achieve the required nanometric resolution.

More information on the overall dynamic design and the other subsystems can be found in [1]. This paper describes the main implementation aspects of the sample positioning system and the related control required for this state-of-the-art X-ray microscopy station. First, the integrated control architecture is reviewed. Next, the high-performance motion control and trajectories are discussed. Finally, commissioning results are presented to demonstrate the current capabilities.

INTEGRATION ARCHITECTURE

Being TARUMÃ a multi-technique end-station that aims at nanometric resolution and high throughput, it requires, besides the dynamic and motion optimization, a high degree of integration between detectors and motion system, as well as a robust system that can perform hundreds of experiments on a daily basis. Past solutions at the 2nd-generation UVX light source at LNLS generally relied on EPICS (Experimental Physics and Industrial Control System) variables to control sample motion and detectors acquisition. Back then, the default update rates up to 10Hz did introduce overhead time in measurements, but those were acceptable due to the lower fluxes available to the beamlines, with correspondingly longer total experiment times. At SIRIUS 4th-generation machine, on the other hand, with photon fluxes up to 1000 fold, the experiment times can be drastically shortened and different types of experiments are required, such that previous solutions become incompatible with TARUMÃ's purposes and capabilities. Therefore, a new solution was proposed and implemented, in which all motion systems and detectors are interconnected via TTL signals that enable fast communication and synchronization among systems, allowing experiments to be up to 130 times faster.

Figure 2 summarizes the integration architecture implemented at TARUMÃ, with the control and detection hardware, and the corresponding signal connections. It is based on the Trigger And Timer Unit (TATU) solution, a flexible in-house development for FPGA-based applications created on NI's CompactRIO (cRIO) [6]. Indeed, capable of receiving leading signals and distributing them to the elements in the control network, such as motion controllers and detectors, it also allows for simple logic operations, customized delays and pulse counting of TTL signals, which can be used for synchronizing motion and data acquisition. Any motion controller with a digital output and capable of reading a trajectory file is suitable as the leading system, depending on the scan routine to be executed. This versatility is what makes the solution easily expandable and general enough to be applied to different beamlines.

At TARUMÃ, the core and most demanding task consists in high-resolution 2D mapping with the XYZ piezo stage. Thus, the standard operation is based on a Main cRIO with an in-house FPGA code that: 1) reads a trajectory file at a 10 kHz rate; 2) translates it into proportional voltage signals that are sent to the XYZ motion controller PI E-727; and 3) provides trigger signals to TATU, running in an independent cRIO,. Hence, TATU's output signals

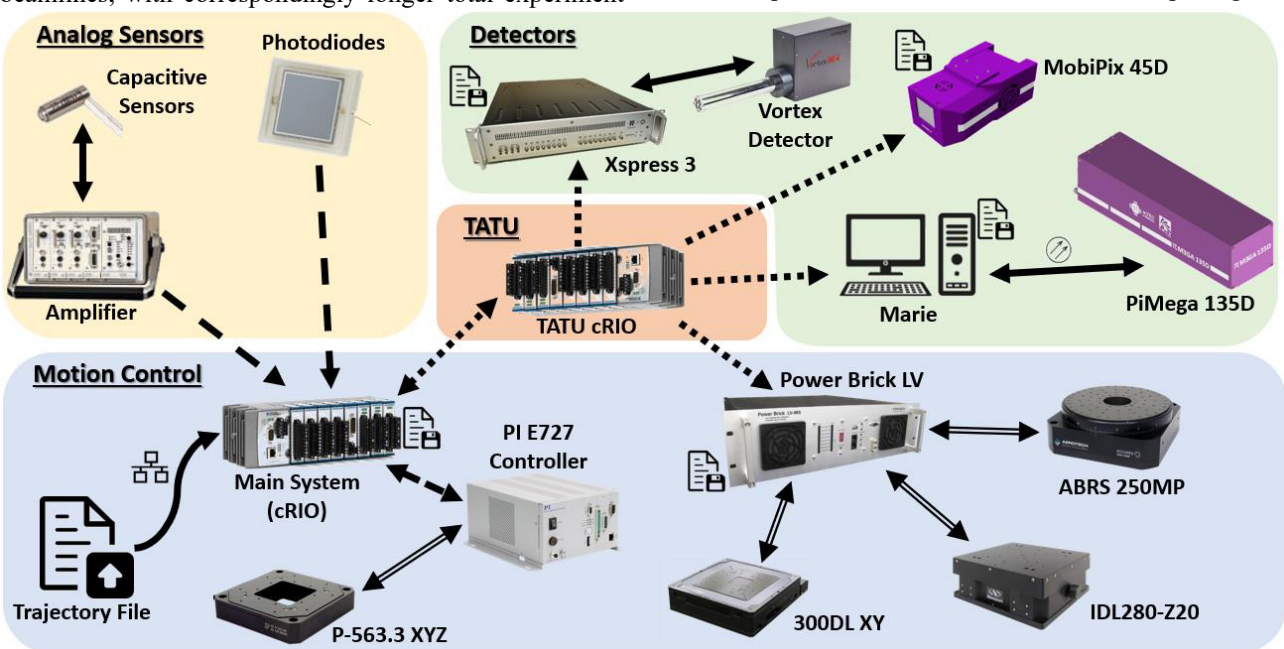


Figure 2: Integration architecture at the TARUMÃ station. The different arrows represent different signal types, namely: solid lines (-) are data transfer signals, dashed lines (--) represent analog signals, dotted lines (.) represent trigger signals and the double thin lines (=) represent closed-loop control signals (feedback and actuators).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

can be used to trigger and synchronize acquisition for: 1) the detectors, with images and energy spectra; 2) the analog signals, which include photodiodes, the capacitive sensors on the metrology frame, and the capacitive sensors in PI P-563.3 stage; and 3) the encoder data from the Omron Delta Tau Power Brick LV (PBLV) sample stage controller, which is further used in 2D and 3D reconstructions.

Nowadays, following a common practice regarding software operation at Sirius beamlines, the fly-scans are planned on a Jupyter Notebook [7], hosted in a Jupyter server that runs on the beamline operation computer. The script is written in Python and allows the user to select: the detectors, the energy regions of interest (ROI) for fluorescence, and the desired trajectory to performed. Currently, 25 predefined trajectories are directly available, as detailed in the following section, whereas unforeseen trajectories may accepted but must be previously calculated and deployed to the cRIO memory. In the near future, the Jupyter Notebook functionality is expected to be expanded with a friendly graphical interface, according to already existing Sirius standards and software tools, in which new trajectories will be dynamically calculated from user input parameters. The Jupyter Notebook is also programmed to configure all detectors and EPICS IOCs desired parameters, such as the number of points and the filenames to be saved.

A fly-scan data-flow example is summarized in Fig. 3. After setting all parameters, the selected trajectory starts with the Main cRIO moving the XYZ piezo stage and sending the trigger signal to TATU, which is properly broadcast to the detectors and complementary controllers. By the end of the task, the files created in each sub-system are merged into one final experiment file and saved in a central storage. Due to the large amount of data generated by the detectors, the files are merged in a powerful Power 8 desktop computer, known by its nickname “Marie”, and then sent to the data storage.

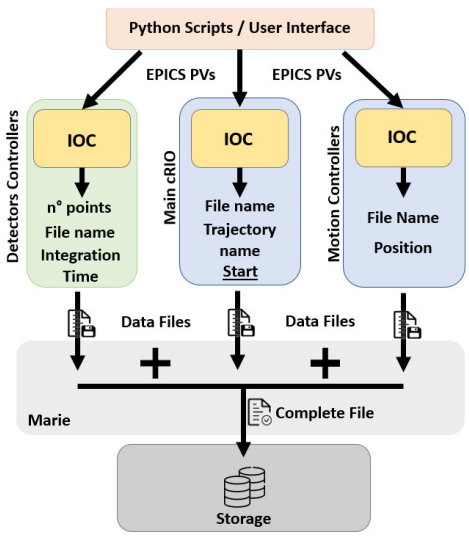


Figure 3: Data-flow schematic in TARUMÃ fly-scans, from the user interface to the data storage.

Still, the flexibility and the easiness in programing the Jupyter Notebook routines in the implemented solution make it possible to include these fly-scans in broader tasks with other beamline components, such as: the sample rotary stage, for tomography; the four-crystal monochromator, for spectroscopy; and the coarse XZ and Y sample stages to perform long-range high-resolution images via stitching. All these different processes have already been successfully commissioned at TARUMÃ, but are also continuously updated and expanded.

MOTION CONTROL

Models and Guidelines

To reach the nanometric positioning capacity, mechanical and control and aspects of the entire station must be taken in count. To help the studies regarding trajectories simple lumped-mass models, as exemplified in figure 4, were implemented to simulate vertical and lateral performances of sample stage and guide motion strategies. The mechanical stiffness between stages and inside the stages (rotor-to-stator) were measured or estimated. In addition, to more accurately derive the models, the individual closed-loop responses were either measured, for the Aerotech DL300 XY and PI 563.3 stages, or estimated, for the IDL280 Y stage, and individually matched in the separate models, before finally integrated.

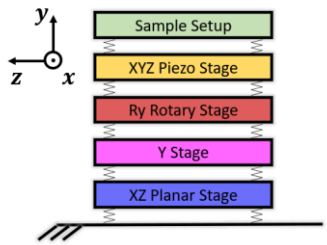


Figure 4: Lumped mass model of the sample stack, used to evaluate vertical and lateral positioning performances.

Firstly, to instruct towards the most suitable trajectory orientation, Fig. 5 depicts the simulated closed-loop Bode plot, both for the vertical and horizontal axes, from the commanded position reference in the P-563.3 to the absolute position of the sample in an arbitrary sample holder weighing 300 g. It can be seen that the vertical axis has a higher bandwidth, around 60Hz, as compared to the horizontal axis, around 20Hz. Thus, the vertical axis exhibits a higher virtual stiffness against disturbances, and can also follow more aggressive trajectory setpoints (velocities and accelerations).

Next, the models were used to investigate the impact of the trajectories on the sample stack, due to, for example, the reaction forces induced in the stages below, with corresponding control errors. Two types of *S-curve* trajectories were considered, namely: 3rd [8] and 4th [9] orders. Third order trajectories are simpler and more commonly employed in commercial motion controllers, but since fourth order trajectories are smoother, due to the higher order continuous derivates, this application was taken as a case study to compare both solutions.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 6 displays the simulated disturbances induced by trajectories of 14.5µm, with 20µm/s cruise velocity, maximum acceleration of 100µm/s^2 and maximum jerk of 1000µm/s^3. The difference lies in the 4th order limiting also the jerk derivate (snap) in 2400µm/s^4. The simulations are for the more compliant horizontal axis for the sake of demonstration, reaching errors in the order of a few nm, but equivalent results in the sub-nm range (scaled roughly by a factor 10) would were found for the vertical axis. Naturally, the magnitude of these errors vary for different accelerations and payloads, but it is possible to notice that the total disturbances induced in the planar stage can be significantly lower and smoother with a 4th-order trajectory, as expected.

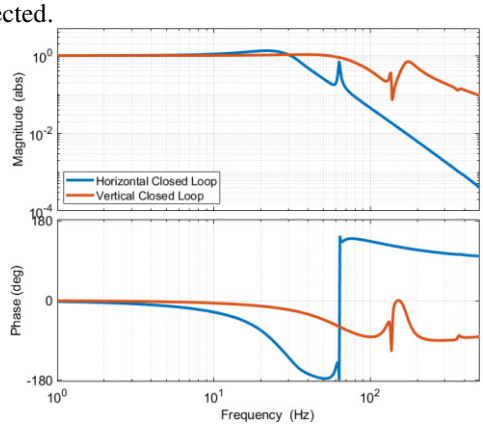


Figure 5: Closed-loop Bode plot for the sample stack in the vertical and horizontal axes, from the P-563 stage reference to the absolute sample position.

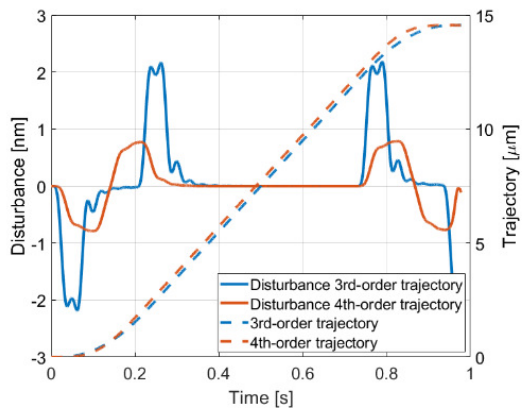


Figure 6: Simulated induced disturbances in the planar 300DL for 3rd and 4th-order trajectories with the P-563.

Trajectories

Based on the previous observations, 4th-order trajectories with the most aggressive setpoints in the vertical axis were chosen for the beamline operation. Then, to cover 2D maps, the fly-scan profile that has been used so far at TARUMÃ is shown in Fig. 7, being composed of long vertical and short horizontal 4th-order trajectories that are concatenated to form a snake-raster pattern. The scans are defined by the following user inputs: vertical velocity, the vertical and horizontal stroke, the trigger length per acquisition point, and the horizontal pitch.

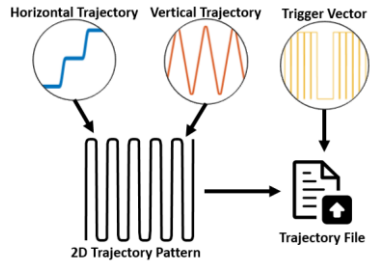


Figure 7: 2D fly-scan pattern at TARUMÃ, highlighting the horizontal and vertical 4th-order trajectories, and the trigger vector as a function of time

To simplify the post-processing of data, the vertical trajectory is calculated so that the constant velocity segment has the exact length of the desired scanning ROI selected by the user. The horizontal trajectory starts when the vertical trajectory is deaccelerating, and it is optimized so that to the horizontal motion ends just before the start of the next constant velocity segment in the vertical axis. This ensures that the accelerations and, consequently, the forces that are applied due to the horizontal motion in the stages below are minimized during acquisition.

The scan vectors are described in 10kHz, which is five to ten times faster than the maximum acquisition rates of 1 to 2kHz that are planned for the beamline. They are saved in the file to be consumed during the experiments. Moreover, besides the vertical and horizontal trajectories, a trigger vector is included in the same file, consisting of a pulse train in which the high level designates acquisition to detectors, such that the acquisition time for each point (integration time) is defined by the pulse width.

Performance refinement

The first offline tests with the piezo XYZ stage indicated following error peaks of up to 500nm for the desired 4th-order vertical trajectories, which might critically compromise data post-processing or at least burden it by requiring more iterations before the final image reconstruction. In that way, the position feedforward gain (*kff*) available in the E-727 controller was studied to optimize the following errors. Figure 8 shows the P-563 stage control error for different scenarios, emphasizing the importance of including an optimal feedforward gain, with improvements by up to a factor 5.

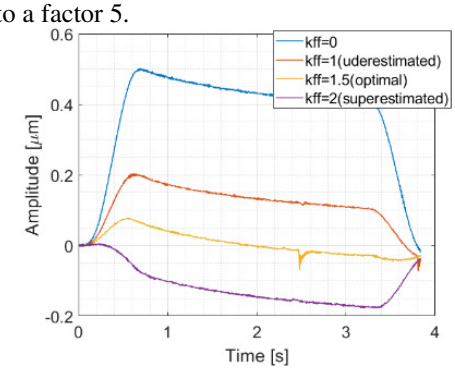


Figure 8: Following errors measured with the P-563 stage for a 4th-order vertical trajectory with different position feedforward gains in E-727 controller.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

Another concern regarding the system controller is the diversity of sample holders that will be used at TARUMÃ, with different weights, and various boundary conditions like gases, vacuum, and temperature. Indeed, keeping control errors and trajectory following performances in the desired levels becomes a practical challenge. The current solution is to previously characterize and optimize the control parameters, including the feedforward gain, for each different setup and provide a selectable controller to the beamline user. For future experiments, a new routine will be developed to automatically optimize (as much as possible) the control parameters for each sample in each sample holder. Such solutions are only possible thanks to the integration of the E-727 controller in EPICS, that allows to easily update the control parameters.

RESULTS

Using the discussed implementation, the first commissioning tests at the beamline, still without beam, were performed during the storage ring 2020/2021 holidays shutdown, whereas the first tests with beam took place in early March 2021. The implementation of the fly-scan routine made it possible to significantly increase the number of experiments performed daily at the CARNAÚBA beamline. For comparison, a 10x10µm^2 mapping, with 10.000 points (100nm resolution) using the previous EPICS-based architecture would take a total of about 2.8 hours, i.e. almost 1s per point. Using the new fly-scan architecture, on the other hand, the same experiment required just 250s before some control optimization and only 75s after that, resulting in a gain of 130 fold for the same quality. Figure 9 shows a florescence image (left) and the ptychography reconstruction (right) of an USAF pattern that is used for beam characterization, validating the motion control and the synchronization between different detectors (for more details, see [2]). The image was acquired in 130s, with a 5x5µm^2 scan and 6.25ms integration time per point.

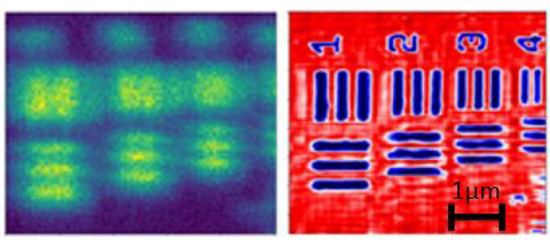


Figure 9: USAF pattern in fluorescence (left) and ptychography (right), with resolution of 300 nm (out of focus) and 70 nm, respectively, validating the synchronism between detectors and motions system.

In addition to throughput, the improvement in experiment time helps minimizing the effects of experimental disturbances, such as vacuum and temperature interlocks, that in the past have been responsible for acquisition losses with the previous architecture. This gain also accelerated several steps in the beamline commissioning, as refinement in mirrors alignment, correction of the KB astigmatism, and commissioning of detectors. Moreover, the minor time per experiment and the modular characteristic of the implementation made it possible to step forward to more advanced experimental routines like energy scans (spectroscopy), tomography, and large 2D mapping (up to 2mm) with sub-micron resolution.

Due to ongoing commissioning and studies regarding the area detectors, the integration time in ptychography and florescence is currently limited to a minimum of 5ms per point, with velocities around 50µm/s to avoid data loss. During aligment, in turn, especial trajectories with 500µs of integration time can be used with transmission data in photodiodes, since the cRIO that acquires the analog signals is capable of measurements up to 100kHz. Yet, to reach sub-ms integration time, velocity must be increased up to 800µm/s, leading to large following errors in addition to making it necessary to rearrange and interpolate the detectors data with respect to the encoders data to compose the final images. Nevertheless, even with interpolation, the quality loss is sufficient for aligment purposes.

As an example, Fig. 10 shows 80x80µm^2 absorption images of a sample with 800nm pixel size, with mapping both in fast scanning with 500µs of integration time (total time of 30s including overhead) in the left, and in standard scanning with integration time of 10ms (total time of 140s including overhead).

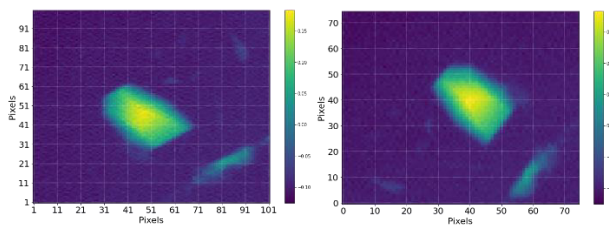


Figure 10: Sample absorption images for interpolated fast mapping at 800µm/s (left) and standard 40µm/s (right).

CONCLUSION

The successful deployment of the first version of fly-scan mapping at TARUMÃ is questions and challenges to a next level. The 100-fold faster scan solution allows the scientists and beamline staff to have immediate answers for samples and the commissioning equipment. Moreover, the general and flexible control architecture enables other motion systems, like the sample coarse stages, or the monochromator, to be integrated to the detectors and perform fast scans as well. The next challenges include the creation of a friendly user interface, and the design of an automatic routine to optimize control parameters for different samples and sample environments. Concerning of more complex experiments, tomography alignment and execution routines and Power Brick fly-scans are under development.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology and Innovation, and the contribution and diligent work of the LNLS teams.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV002

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] R. Geraldes et al., "Design and Commissioning of the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at. MEDSI 2020, Chicago, USA, Jul. 2021, paper WEPB13, unpublished.

[2] Tolentino, H.C.N., et al., "X-ray microscopy developments at Sirius-LNLS: first commissioning experiments at the Carnauba beamline", in *Proc*. SPIE X-Ray Nanoimaging: Instruments and Methods , San Diego, California, USA, Sep. 2021, doi: 10.1117/122596496.

[3] Sirius Project, https://www.lnls.cnpem.br/sirius-en/.

[4] W. H. Wilendorf et al., "Electrochemistry and Microfluidic for the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at. MEDSI 2020, Chicago, USA, Jul. 2021, paper WEPC03, unpublished.

[5] F. R. Lena et al., "A Cryogenic Sample Environment for the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at. MEDSI 2020, Chicago, USA, Jul. 2021, paper WEPC02, unpublished.

[6] J. R. Piton et al, "TATU: a flexible FPGA-based trigger and timer unit created on CompactRIO for the first SIRIUS beamlines", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Beijing, China, Oct 2021, paper THPV021, this conference.

[7] Jupyter Notebook, https://jupyter.org/.

[8] P. Lambrechts, "Trajectory planning and feedforward design for electromechanical motion systems", TUE, Eindhoven, Netherlands Rep. DTC 2003-18, Apr. 2003.

[9] Advanced Setpoints for Motion Systems, https://www.mathworks.com/matlabcentral/fileexchange/16352-advanced-setpoints-for-motion-systems

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

THE DYNAMIC MODELING AND THE CONTROL ARCHITECTURE OF THE NEW HIGH-DYNAMIC DOUBLE-CRYSTAL MONOCHROMATOR (HD-DCM-Lite) FOR SIRIUS/LNLS

G. S. de Albuquerque*, A. V. Perna, J. L. Brito Neto, M. A. L. Moraes, M. S. Souza, M. Saveri Silva, R. R. Geraldes[1], LNLS, Campinas, Brazil
[1]also at the CST Group, Eindhoven University of Technology (TUe), Eindhoven, The Netherlands

Abstract

The High-Dynamic Double-Crystal Monochromator (HD-DCM) has been developed since 2015 at Sirius/LNLS with an innovative high-bandwidth mechatronic architecture to reach the unprecedented target of 10 nrad RMS (1 Hz - 2.5 kHz) in crystals parallelism also during energy flyscans. Now, for beamlines requiring a smaller energy range (3.1 keV to 43 keV, as compared to 2.3 keV to 72 keV), there is the opportunity to adapt the existing design towards the so-called HD-DCM-Lite. The control architecture of the HD-DCM is kept, reaching a 20 kHz control rate in NI's CompactRIO (cRIO). Yet, the smaller gap stroke between crystals allows for removing the long-stroke mechanism and reducing the main inertia by a factor 6, not only simplifying the dynamics of the system, but also enabling faster energy scans. With sinusoidal scans of hundreds of eV up to 20 Hz, this creates an unparalleled bridge between slow step-scan DCMs, and channel-cut quick-EXAFS monochromators. This work presents the dynamic error budgeting and scanning perspectives for the HD-DCM-Lite, including discussions on the feedback control via loop shaping, feedforward considerations, and leader-follower control strategies.

INTRODUCTION

After successfully designing, installing and commissioning the High-Dynamic Double-Crystal Monochromator (HD-DCM) [1] at the MANACÁ (crystallography) and the EMA (extreme conditions) beamlines, QUATI (quick absorption spectroscopy) and SAPUCAIA (small-angle-scattering) are two forthcoming Sirius beamlines demanding an HD-DCM at the Brazilian Synchrotron Light Laboratory (LNLS). Since these new beamlines require a smaller energy range (3.1 to 43 keV), the total gap stroke of the instrument can be significantly reduced from 9 mm to about 2.75 mm, such that an opportunity is created to adapt the existing design towards the so-called HD-DCM-Lite.

Removing the long-stroke module (see [1]) for the large gap adjustments allows not only for cost reduction and simplification in the assembly, but also for significant improvement in dynamics. By reducing the main inertia by a factor of 6, the HD-DCM-Lite is expected to deliver energy flyscans of hundreds of eV up to at least 4 to 40 times per second, while keeping fixed exit and the inter-crystal parallelism in the range of a few tens of nrad RMS (root mean square) for the 1 Hz - 2.5 kHz range. Thus, QUATI (superbend-based) may take full advantage of this capability in time-resolved analysis, whereas new science opportunities may be explored for SAPUCAIA. The new design focuses on extending the scanning capabilities while preserving positional accuracy, creating a solution between the current HD-DCM (limited in speed but with fixed-offset and already extremely stable) and fast channel-cut monochromators [2], which suffer from beam walk due to offset variation.

An overview of the mechanical design of the HD-DCM-Lite, with the CAD drawing of its in-vacuum mechanics and a simplified lumped-mass model, is given in Fig. 1. Two crystal sets, with Si(111) and Si(311) orientations for options in energy filtering bandwidth and range, are mounted side by side, being alternatively selected during operation. At the lower side, the so-called 1st Crystals (CR1) (6) are mounted to a common Metrology Frame (MF1) (5), which, in turn, is fixed to an Auxiliary Frame (AF1) (4), that is, finally, fixed to the Goniometer Frame (GOF) (3), which is driven by two rotary stages (ROT) (2) for controlling the angle of incidence of the incoming X-ray beam on the crystals for energy selection according to Bragg's law of diffraction.

At the upper side, the so-called 2nd Crystals (CR2) (7) are used to redirect the beam in the fixed-exit configuration that defines the operational principles of an X-ray DCM. They are mounted to a common metrology frame as well, which, in this case, is a Short-Stroke stage (SHS) (8) that is actively controlled with nanometer level with respect to the MF1 in 3 degrees of freedom (DOF). Indeed, the distance (gap) and parallelism between crystals — or, more precisely, between their metrology frames — can be adjusted via closed-loop control based on 3 voice-coil actuators and 3 laser interferometers. The reaction forces of these actuators act on a Balance-mass (BMS) (10) for dynamic filtering purposes, and both the SHS and the BMS are mounted via flexural folded leaf-springs to the Auxiliary frame 2 (AF2) (9), which is also fixed to the GOF.

The supports of the ROT are stiffly fixed to a vacuum flange, as part of the complete vacuum vessel (VES) (not shown), which, in turn, is stiffly fixed to a granite bench (GRA) (1) for alignment purposes and crystal set selection at the beamline, building the dynamic architecture of the system all the way up from experimental floor (GND).

With the previous experience from the HD-DCM as a high-end mechatronic system, here again a systematic approach based on precision engineering principles and predictive modeling has been adopted to maximize the efficiency in development time and costs, according to a "first-time-right"

* guilherme.sobral@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

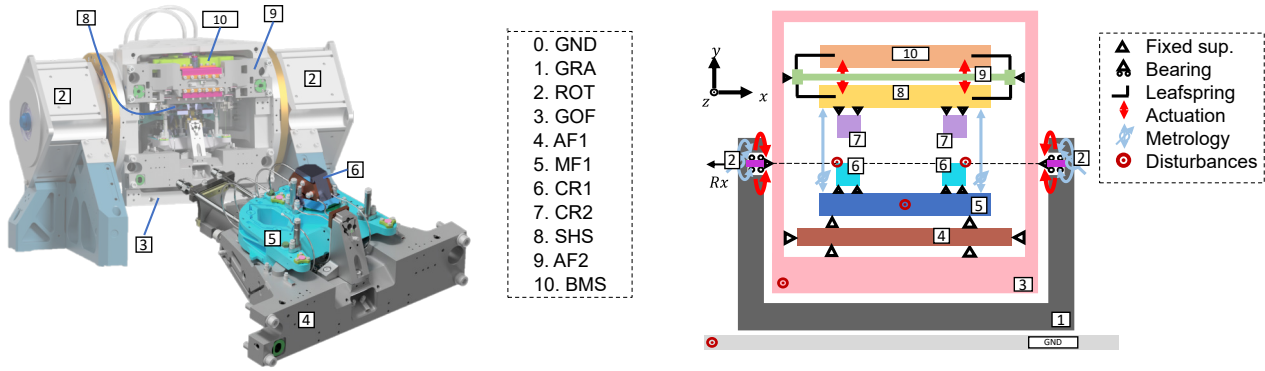


Figure 1: Left: HD-DCM-Lite in-vacuum main parts. Right: Lumped-mass model including actuators, sensors and disturbances, as used for dynamic error budgeting performance predictions.

philosophy. Using the Dynamic Error Budgeting (DEB) methodology [3, 4], the effects of system disturbances can be investigated as the mechatronic design of the machine is iteratively refined until compliance with specs.

Introductory aspects of the HD-DCM-Lite, such as the control effort, maximum displacement amplitude, speed budget, and thermal management, and the resulting time-resolved capabilities analysis were briefly introduced in [5]. Regarding its control architecture, it is expected to very closely follow the FPGA-based solution with NI's CompactRIO (cRIO) that has been developed for the HD-DCM, such that control rates up to 20 kHz should be possible for conformity with closed-loop bandwidths up to 250 Hz, or higher. The latest status in this topic, including integration aspects at Sirius beamlines can be found in [6].

Here, using a DEB toolbox that has been developed at LNLS during the HD-DCM project, the following sections discuss the mechatronic and disturbance models for the system, and present the high-performance predictions both in stand-still and scanning operation modes.

MECHATRONIC MODEL

As described below, the mechatronic model consists in a state-space representation composed of the electro-mechanical plant, including the mechanical dynamics, actuators and sensors, and the controllers.

Plant

The mechanical plant is developed according to the lumped-mass model shown in Fig. 1, with the mechanical system being derived in 6DOF via Dynamic Substructuring (DS) methods [7]. Firstly, inertia and geometrical parameters for each body are extracted from the CAD drawing in Autodesk® Inventor and exported to MATLAB® , where the whole customized toolbox has been implemented. Then, the links between lumps are defined as stiffness values, obtained either from simulated Structural Analysis with Ansys® Mechanical or from experimental data (see [3]). Next, the mechanical system is coupled via well-known DS transformations, written in state-space representation and put in modal basis, such that modal damping factors, based on experience or experimental results, can be individually applied to each mode shape. Finally, transformation matrices are computed to account for the geometry of the metrology points of sensors, and the action points of actuators and points subject mechanical disturbances.

Controllers

As the HD-DCM, the HD-DCM-Lite was designed in such way that the DOF of interest can be statically decoupled to transform the original multiple-input-multiple-output (MIMO) system in independent single-input-single-output (SISO) systems, which greatly simplifies the design of the controllers and robustness analyses [8]. Thus, four individual control loops can be considered, namely: BRG, for the Bragg angle; and GAP, PTC and RLL, for the inter-crystals gap, and the pitch and roll angles, in the so-called crystal cage (CCG), respectively.

Feedback The first step consists in designing robust feedback controllers, which is done via the loop-shaping technique [9]. Indeed, thanks to the damped mass-spring approximation (second-order system) with low-frequency decoupling that characterizes the plants of the four loops (see [8]), simple and generic proportional-integral-derivative (PID) controllers can be implemented with practical "rules of thumb" [10].

First, the bandwidth frequencies, defined as those in which the open-loop transfer functions cross 0 dB, are chosen as $f_{bw} = 20$ Hz for the BRG and $f_{bw} = 250$ Hz for the CCG loops. Then, for phase margin, a lead filter is added with a zero at $f_{bw}/3$ and a pole at $3f_{bw}$. Next, to avoid high-frequency amplification of dynamics and reduce sensor noise, a low-pass filter is added at $5f_{bw}$. After that, an integral action is added with a zero at $f_{bw}/5$ for zero-frequency (DC) error. Finally, analyzing the open-loop system for gain and phase margins, a couple of notch filters may be added as necessary. Here, 2 notches are used for the RLL loop and 1 for the remaining ones. The modeled feedback closed-loop

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

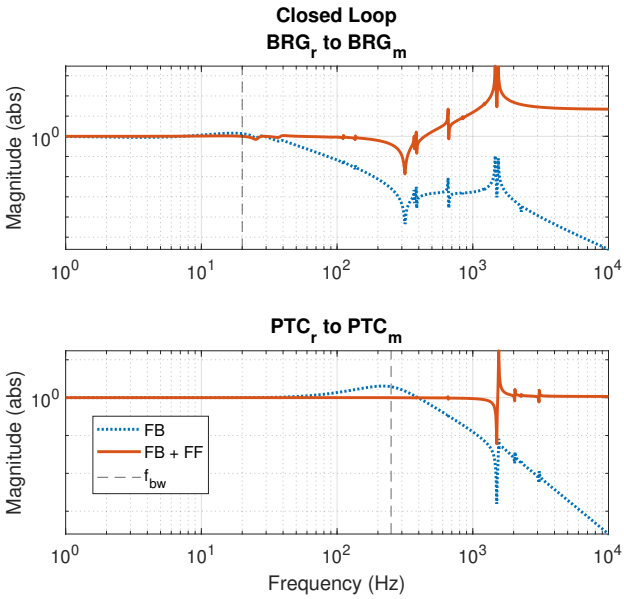


Figure 2: BRG and PTC transfer functions for the feedback closed-loop and for a feedback plus feedforward equivalent. The feedback bandwidths are also indicated in dashed lines.

transfer functions from reference (r) to output (m) for BRG and PTC can be seen in Fig. 2.

Feedforward To reduce the transient errors and expand the demanding scanning possibilities beyond the feedback bandwidth limits, particularly for the BRG loop, simple feedforward filters are also included in the model. Thus, assuming well-designed trajectory setpoints, with limited energy content in the high-frequency range, and once again benefiting from the reasonable second-order approximation for the plants in the low and mid-frequency ranges, high-order inversion-based filters can be prevented, and the feedforward blocks can be simply based on stiffness, damping and inertia constants.

In the model, the parameters can be directly extracted and perfectly compensated for. In practice, experimental tuning shall be realized. In this sense, this simplified feedforward structure will also ease the experimental tuning, thanks to the small number of parameters, in addition to saving implementation resources in the control hardware.

As an indication of the intended concept, the simulated transfer functions from reference (r) to output (m) with feedback plus feedforward for BRG and PTC are also depicted in Fig. 2. It can be seen that setpoints over a broader frequency range should be possible thanks to the feedforward action, but also that the energy content of these setpoints (visible in Fourier analyses, for instance) should be carefully designed below 200 Hz to prevent issues, such amplification effects and saturation, especially in the BRG loop.

DISTURBANCE MODELS

Once the mechatronic model has been built, the disturbances that will eventually determine the system performance must be addressed. Conceptually, they may be grouped as mechanical disturbances, as forces acting on the system, and electronic disturbances, which are related to the control hardware.

The main mechanical disturbances are the floor vibrations (GND) and Flow-Induced Vibrations (FIV) induced by the cryogenic cooling architecture of the silicon crystals via liquid nitrogen (see [11]). For the GND, the floor spectra is directly available from measurements with accelerometers and seismoemeters. Modeling FIV disturbances, on the other hand, is far more challenging due to the complexity of the flow effects and the cryogenic conditions required for real measurements. Thus, FIV was modeled according to best guesstimates. Theses disturbances are described in details for the HD-DCM in [3] and reused here.

The electronic disturbances, in turn, can be subdivided into actuators and sensors. All actuators will be controlled via the NI-9269 digital-to-analog converter (DAC), with a certain contribution in electronic noise, such as quantization. Then, the Aerotech APR260-S-240-83 rotary stages will use the Varedan LA-415-SA-T linear amplifier, whereas the Akribis AVM40-HF-6.5 voice-coils will rely on Trust Automation TA105. Each amplifier has its own noise level and spectral signature, affecting the closed-loop control error. Regarding the sensors, the rotary stages will be based on Renishaw TONIC encoders, whereas the sensors in the CCG will be SmarAct PicoScale Michelson interferometers. This electronic noise also enters the closed-loop system as disturbances. All these elements were already experimentally characterized for HD-DCM, as detailed in [3] as well.

SDE

While, as demonstrated in [3], the previous disturbance sources are sufficient to predict and describe the HD-DCM and the HD-DCM-Lite performances in stand-still condition, another type of sensor disturbance is introduced during motion, which becomes critical for the fast scanning purposes of the HD-DCM-Lite. Indeed, Sub-divisional Errors (SDE) are periodical patterns that are commonly found in optical sensors, such as encoders and interferometers, resulting from the measurement physical principles, together with alignment characteristics, and each particular electronic processing [12].

Based on empirical data, the SDE of the interferometers were found to be in the order of ± 2 nm with a periodicity of 775 nm (i.e. half of its infrared wavelength), which has a negligible impact on the control error of the system. The SDE for rotary encoder, on the other hand, reaches 300 nrad of amplitude with a periodicity of 10 µm (i.e. half of the scale pitch), which translates to about 100 µrad given the scale size. Since the typical stand-still control errors in the rotary stages are in the range of few tens of nrad RMS (see Table 1 in the Performance Predictions section), the harmful potential of the encoder SDE becomes clear.

Being artifacts, these patterns are deviations from the ideal measurements, and intrinsic accuracy limitations in the sensors, which can not be differentiated by the control

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

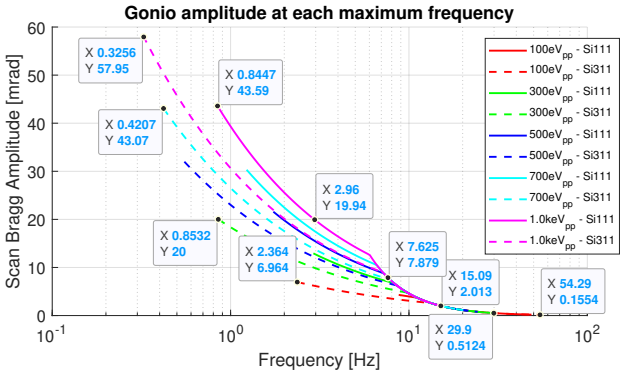


Figure 3: Boundary scanning possibilities for different energy scan amplitudes with either crystal set and ten representative points for performance analysis.

loop *a priori* from what would be the ideal signal. Thus, during motion, unless more advanced calibration and filtering strategies can be implemented via control, these errors appear at frequencies related to the traveling speed, being introduced in the closed control loop through its complementary sensitivity function.

Hence, at low frequencies the errors are followed by the system, around the bandwidth frequency they are amplified, and then they are gradually filtered out for increasingly higher frequencies. In the following and amplification regions these errors are converted to actuation effort (torque), which may excite internal mechanical resonances or at least increase the disturbances to the CCG control loops. This is why this effect must be carefully investigated.

Still, as the encoder SDE contribution depends completely on the trajectory being executed, some assumptions must be made for modeling purposes. Considering that as many repetitive energy scans per second as possible are desired for spectroscopy experiments with the HD-DCM-Lite, torque and motion disturbances can be expected to be minimized for sinusoidal trajectories in the rotary stage.

Taking into account all practical boundary limitations, and the non-linear relations among the desired energy and the Bragg angle for both crystal sets and gap between crystals, Fig. 3 provides an abacus with maximum expected scanning frequency for the Bragg amplitudes related to different energy scan amplitudes (see [5]). Finally, given the sinusoidal trajectories, the SDE can be modeled via *sine sweeps*, in which the maximum frequency is proportional to the maximum sinusoidal angular speed. Ten case studies selected for performance analysis in the following section are also highlighted in Fig. 3.

PERFORMANCE PREDICTIONS

Having the mechatronic and disturbance models, the performance predictions can be finally derived in the DEB toolbox. Thanks to the superposition nature of the linear time-invariant (LTI) formalism, the role of the different error sources can be independently investigated, which, in particular, allows for straightforward comparisons between static and scanning performances. Since the low frequency content of the reference trajectories should be mostly handled by feedforward, the SDE becomes the main additional noise source during scans.

A graphical example of one such performance prediction is illustrated in the cumulative power spectrum (CPS) plots in Fig 4, with the disturbances grouped according to what was presented in the Disturbances section. It shows the leftmost case of Fig. 3, i.e. sinusoidal scans with the largest Bragg amplitude (around 60 mrad, or 3.4 deg) and lowest sinusoidal scanning frequency (around 0.3 Hz). It can be seen that the SDE has a brutal effect in the BRG loop, being more than 10 times larger than the second highest effect, which is the GND. The total error approaches 0.5 µrad, which exceeds the original spec of 0.15 µrad by more than a factor 3, requiring some attention, as discussed below.

Similarly, for the PTC, which is the most sensitive control loop for beam position variation, the SDE dominates the error budget, with the 10 nrad spec being marginally exceeded. Without it, representing the stand-still condition, the errors are limited to about 5 nrad only. For the GAP and RLL loops, in turn, most of the contribution is due to the floor vibrations, with negligible contributions from the remaining elements. Still the values are well within the specs of 300 nm and 90 nrad, respectively.

The complete prediction results, with the RMS (1 Hz-2.5 kHz) control error for all simulated sinusoidal scan trajectories, are summarized in Table 1. The stand-still subset and original specification are also indicated in the last rows for completeness. It should be noted that the frequency and angular amplitude are identifying the scans according to Fig. 3, but these two parameters alone are not sufficient to fully describe a scan, since the mean energy also plays a role. Nonetheless, it is possible to notice that the SDE influence grows with the scanning frequency, as it might be expected.

It can be seen the the BRG loop is the one that is most critically affected by the SDE, with the errors exceeding the original spec by more than a factor 10. Still, the PTC also shows increasing errors with higher scanning frequencies, reaching nearly 40 nrad. Thus, a future topic of research is looking for a practical solution for filtering the sweeping SDE contribution via control or calibration. In the meantime, if truly sensitive to these error levels, the experiments may need to be limited to sufficiently lower scanning frequencies.

Furthermore, it is worth mentioning that, while the CCG loops are practically implemented as followers of the BRG loop, due to the geometrical relation of the gap and to fine calibrations with respect to the Bragg angle (see [13]), the appropriate leading signal must come from the BRG reference, not its sensor (see also [6]). All the performance results presented here follow this approach. Indeed, the control errors in the BRG are small as far as the gap and the calibration parameters are concerned, such that the reference signal provides a sufficiently accurate reference for the remaining loops. Otherwise, if following the measurement signal, the whole dynamics of the BRG loop, including its

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

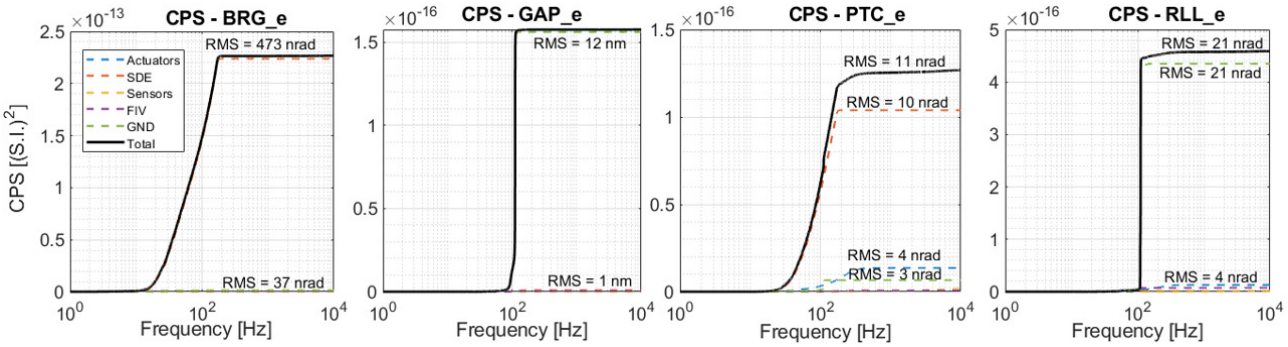


Figure 4: Cumulative power spectrum (CPS) according to the DEB simulations considering all the disturbance sources for a sinusoidal scan with $f = 0.3$ Hz and $\theta_A = 60$ mrad.

Table 1: Position stability (RMS) results for disturbances on control loops for different scan amplitudes θ_A and frequencies f between 1 kHz to 2.5 kHz

θ_A [mrad]	f [Hz]	BRG_e [nrad]	GAP_e [nm]	PTC_e [nrad]	RLL_e [nrad]
60	0.3	470	12	11	21
40	0.4	450	12	11	21
40	0.9	430	12	9.6	21
20	0.9	460	12	11	21
20	3.0	780	12	13	21
7.0	2.4	1000	12	22	21
8.0	7.7	1600	12	24	22
2.0	15	1800	13	36	22
0.5	30	1300	13	30	22
0.2	54	1600	13	38	22
No SDE:		55	12	4.8	21
Spec:		150	300	10	90

SDE errors would couple to the CCG loops, more significantly degrading their performances.

The scanning results discussed here find their most direct potential at bending magnet beamlines, such as QUATI, due to their broad-band emission spectra. With undulators, opportunities will depend on motion limitations, including the potential influences on the storage ring, and detuned operation possibilities. Hence, in the short term, integration and synchronization efforts are expected to be focused mainly between the monochromator and the detectors (which may reach acquisition rates of a few MHz). In that sense, promising initial results have been recently demonstrated with the HD-DCM (see [14] and [15]).

Finally, one potential disturbance source that may turn out to have a significantly negative impact in scans is the mechanical bearings in the rotary stage. The reason why it has been left out of the analyses, was the difficulty in modeling it, given the strong non-linear dependencies on mechanical tolerances, payload, speed and lubrication/friction. Consequently, although conservative in the modeled disturbances, the numbers presented here cannot be considered fully conservative performance predictions for the HD-DCM-Lite design. The commissioning results with the first prototype, due to the next few months, will provide further insights and occasionally suggest that smoother bearings, such as contactless air bearings or magnetically levitating bearings, may be required for ultimate performance.

CONCLUSION

Dynamic error budgeting (DEB) is a powerful methodology to be used in the predictive design of high-performance mechatronic systems. The toolbox developed in-house over the course of the High-Dynamic Double-Crystal Monochromator (HD-DCM) project is now applied in the development of the HD-DCM-Lite for extended scanning capabilities at Sirius light source. The model predictions, which provided very good agreement with experimental data in the HD-DCM, here point to an unprecedented inter-crystal parallelism level in the range of 5 nrad RMS (1 Hz – 2.5 kHz) in stand-still conditions. Yet, the results also suggest that the detrimental effects of sub-divisional errors (SDE) in the sensors to the most demanding scanning trajectories might increase the control errors by more than a factor 10. The actual impact of the occasional performance reduction may need to be confronted with particular experimental sensitivity at the beamlines, but effective filtering strategies should be investigated as soon as possible. Finally, possibly limiting disturbances arising from the mechanical bearings during scans will require deeper experimental investigations during commissioning, which is expected by early 2022.

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology and Innovation and the contributions of the LNLS team. They also recognize the essential role of MI-Partners in the development of the original HD-DCM and in the workflow adopted here, and the discussions that allowed for the development of many of the tools used in this work as part of a PhD project carried out with the CST group at TUe, with Marteen Steinbuch and Hans Vermeulen as promoters, and Gert Witvoet as co-promoter.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV003

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] R. Geraldes *et al.*, "The Status of the New High-Dynamic DCM for Sirius," en, *Proceedings of the Mechanical Eng.~Design of Synchrotron Radiation Equipment and Instrumentation*, vol. MEDSI2018, 6 pages, 1.106 MB, 2018, Artwork Size: 6 pages, 1.106 MB ISBN: 9783954502073 Medium: PDF Publisher: JACoW Publishing, Geneva, Switzerland. DOI: 10.18429/JACOW-MEDSI2018-WEOAMA01. http://jacow.org/medsi2018/doi/JACoW-MEDSI2018-WEOAMA01.html

[2] O. Müller, D. Lützenkirchen-Hecht, and R. Frahm, "Quick scanning monochromator for millisecond *in situ* and *in operando* X-ray absorption spectroscopy," en, *Review of Scientific Instruments*, vol. 86, no. 9, p. 093 905, Sep. 2015, ISSN: 0034-6748, 1089-7623. DOI: 10.1063/1.4929866. http://aip.scitation.org/doi/10.1063/1.4929866

[3] R. R. Geraldes, M. A. L. Moraes, R. M. Caliari, and G. Witvoet, "Dynamic Error Budgeting in the development of the High-Dynamic Double-Crystal Monochromator for Sirius light source," *American Society for Precision Engineering Topical Meetings*, 2020.

[4] L. Jabben and J. van Eijk, "Performance analysis and design of mechatronic systems," *Mikroniek - Professional Journal on Precision Engineering*, vol. 51, no. 2, pp. 5–11, 2011, bibtex[publisher=Dutch Society for Precision Engineering].

[5] A. V. Perna *et al.*, "The HD-DCM-Lite: A High-Dynamic DCM with Extended Scanning Capabilities for Sirius/LNLS Beamlines," in *presented at the 11th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'20)*, JACoW Publishing, Jul. 2021.

[6] R. R. Geraldes *et al.*, "The fpga-based control architecture, epics interface and advanced operational modes of the high-dynamic double-crystal monochromator for sirius/lnls," in *presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, (Shanghai, China), JACoW Publishing, Oct. 2021.

[7] D. de Klerk, D. J. Rixen, and S. N. Voormeeren, "General Framework for Dynamic Substructuring: History, Review and Classification of Techniques," en, *AIAA Journal*, vol. 46, no. 5, pp. 1169–1181, May 2008, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.33274. https://arc.aiaa.org/doi/10.2514/1.33274

[8] R. M. Caliari, R. R. Geraldes, M. A. L. Moraes, and G. Witvoet, "Loop-Shaping Controller Design in the Development of the High-Dynamic Double-Crystal Monochromator at Sirius Light Source," *American Society for Precision Engineering Topical Meetings*, 2020.

[9] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*, 2nd ed. Hoboken, NJ: John Wiley, 2005, ISBN: 978-0-470-01167-6.

[10] R. M. Schmidt, G. Schitter, and J. v. Eijk, *The design of high performance mechatronics: high-tech functionality by multidisciplinary system integration*, English. 2020, OCLC: 1145888837, ISBN: 978-1-64368-051-4. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2401172

[11] R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius," en, *Proceedings of the 9th Edition of the Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference*, vol. MEDSI2016, 6 pages, 0.966 MB, 2017, Artwork Size: 6 pages, 0.966 MB ISBN: 9783954501885 Medium: PDF Publisher: JACoW Publishing, Geneva, Switzerland. DOI: 10.18429/JACOW-MEDSI2016-TUCA05. http://jacow.org/medsi2016/doi/JACoW-MEDSI2016-TUCA05.html

[12] A. Ellin and G. Dolsak, "The design and application of rotary encoders," en, *Sensor Review*, vol. 28, no. 2, J. Billingsley, Ed., pp. 150–158, Mar. 2008, ISSN: 0260-2288. DOI: 10.1108/02602280810856723. https://www.emerald.com/insight/content/doi/10.1108/02602280810856723/full/html

[13] R. R. Geraldes *et al.*, "Commissioning and Prospects of the High-Dynamic DCMs at Sirius/LNLS," in *presented at the 11th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'20)*, JACoW Publishing, Jul. 2021.

[14] R. R. Geraldes *et al.*, "Design and Commissioning of the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS," in *presented at the 11th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'20)*, JACoW Publishing, Jul. 2021.

[15] J. R. Piton *et al.*, "Tatu: A flexible fpga-based trigger and timer unit created on compactrio for the first sirius beamlines," in *presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, (Shanghai, China), JACoW Publishing, Oct. 2021.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV005

EXPERIMENT AUTOMATION USING EPICS

D. Cosic[†1], M. Vićentijević, Ruđer Bošković Institute, Zagreb, Croatia
[1]Faculty of Electrical Engineering, Mechanical Engineering and Navel Architecture, University of Split, Split, Croatia

Abstract

Beam time at accelerator facilities around the world is very expensive and scarce, prompting the need for experiments to be performed as efficiently as possible. Efficiency of an accelerator facility is measured as a ratio of experiment time to beam optimization time. At RBI we have four ion sources, two accelerators, ten experimental end stations. We can obtain around 50 different ion species, each requiring a different set of parameters for optimal operation.

Automating repetitive procedures can increase efficiency of an experiment and beam setup time. Currently, operators manually fine tune the parameters to optimize the beam current. This process can be very long and requires many iterations. Automatic optimization of parameters can save valuable accelerator time. Based on a successful implementation of EPICS [1], the system was expanded to automate reoccurring procedures. To achieve this, a PLC was integrated into EPICS and our acquisition system was modified to communicate with devices through EPICS. This allowed us to use tools available in EPICS to do beam optimization much faster than a human operator can, and therefore significantly increased the efficiency of our facility.

INTRODUCTION

Some experiments performed have a standardized procedure which must be repeated on many samples sequentially. In the Laboratory for Ion Beam Interaction at the Ruđer Bošković Institute these experiments are for material analysis purposes and are performed on a dedicated beam line specifically designed for such studies. Materials are studied by applying Rutherford Backscattering (RBS) and Particle Induced X-Ray Emission (PIXE) ion beam analysis techniques. Using these quantitative analysis methods, a precise elemental composition (N_Z) of the target samples can be calculated. Typically, the elemental concentrations of an unknown sample are determined by measuring a standard with a known concentration of an element and comparing it to the sample of interest, using the following formula [2].

$$N_Z = N_A \frac{I_Z m_Z \sigma_A^X eff_A Q_A}{I_A m_A \sigma_Z^X eff_Z Q_Z} \quad (1)$$

where N_Z is the concentration (g/cm^2) of the element in the sample, N_A is the concentration in the standard, I_Z and I_A refer to the integral of the peak in the accumulated spectrum, m is the mass, σ is the cross-section, eff is the efficiency of the detectors, and Q is the integrated projectile charge. The experimental chamber on this beam line consists of two particle detectors and two x-ray detectors that are connected to a custom data acquisition system [3]. The integral of the peak (I) is calculated by constructing a histogram from the data acquired from the detectors. The total charge (Q) is collected by measuring the current from the target if it is a thick sample, or a Faraday cup located behind the sample, if it is thin. This current is measured by an Electrometer and integrated over the time of the experiment to get the total projectile charge on the target.

The measurement procedure consists of the following steps:

1. Setting the target in the beam position.
2. Setting the acquisition system to start recording.
3. Resetting the charge counter.
4. Opening the chamber valve so the ion beam can enter the chamber and hit the target.
5. Waiting for enough statistics to be collected for a well-defined histogram, which typically requires about 1 μC of charge.
6. Closing the valve to the chamber.
7. Stopping the acquisition system.
8. Reading the total charge from the electrometer.
9. Logging the results.

These 9 steps must be repeated for each sample being measured which in one day could be as many as 30 times. Due to this highly repetitive process, errors can occur where the electrometer is not reset, the positioning of the sample is not correct or simply the steps are preformed out of order. Such mistakes result in having to repeat measurements, decreasing beam time utilization efficiency.

Many of these steps can be automated if the various hardware and software components could be controlled remotely. EPICS was used to interconnect all the elements because it creates a common communication framework through which repetitive processes can be automated.

IMPLEMENTATION

As we are still in the preliminary phase of upgrading our accelerator control system to an industrial standard such as EPICS, it was crucial that all implementation for this work were made to work in parallel with our existing system and could easily be disconnected. The existing system utilized

† email address: dcosic@irb.hr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

a local Ethernet control network which communicates with all the instrumentation in the facility through a PLC, serial routers or directly TCP protocols. All instrument control is performed from a single computer running LabView software which sets and reads variables through this control Ethernet network. An EPICS server was connected to this existing Ethernet control network and two soft Input Output Controllers (IOCs) were programmed to scan parameters from instruments on the network without writing to them, leaving the current LabView based control system to run without interference.

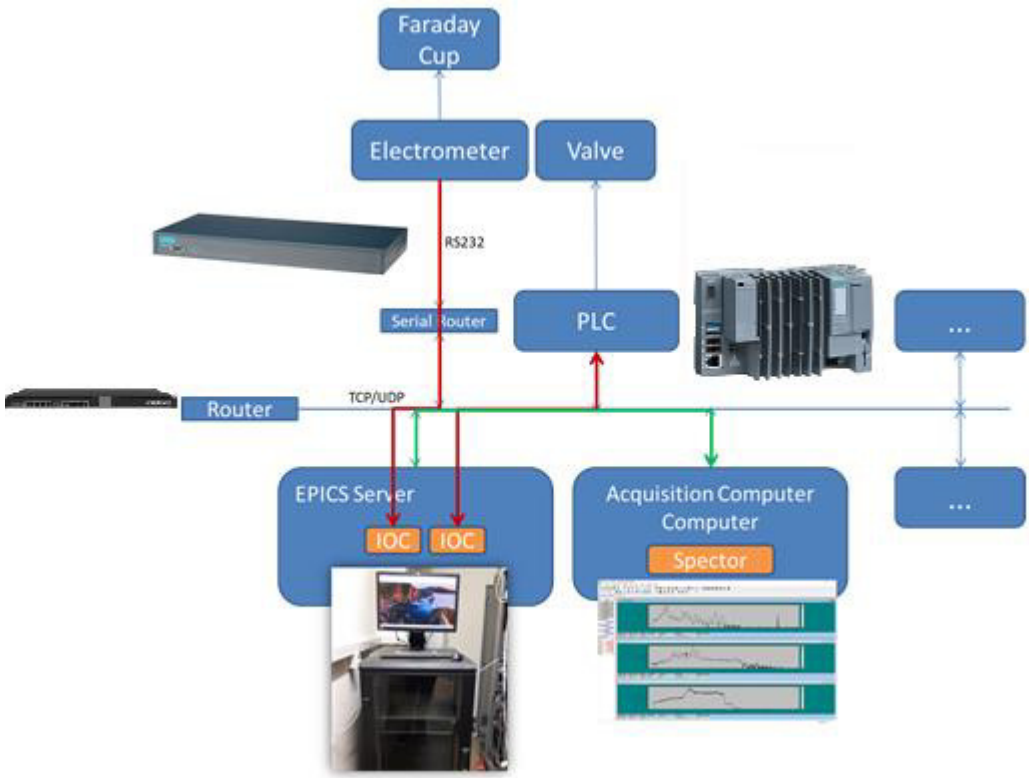


Figure 1: System Implementation.

Illustrated in Figure 1, the EPICS server was connected to the Ethernet control network and connected to the Faraday cup located on the beam line, the pneumatic valve before the chamber and the acquisition software running on the acquisition computer using the EPICS framework. A dedicated IOC was created to communicate with the Keithley 6514 System Electrometer through a virtual COM port provided by an Advantech EKI-1526 serial router. The IOC records were configured to abstract three commands to the network. The first command initializes the instrument, changes the operation mode to charge measurement and then sends zero set and zero correct instructions to the instrument. The second command reads the current charge reading from the instrument while the third command estimates the ion beam current by monitoring the charge change over a constant period of time. A second IOC was created to control the valve which is connected to a SIEMENS SIMATIC ET 200SP PLC. This IOC contains two commands. One to get the current valve position and a second that sets the valve position. The IOC reads and writes to the variable database inside the PLC to control the valve.

Instead of using a EPICS sequencer [4], communication with the abstracted EPICS commands was implemented into the acquisition control system used on the beam line. The data acquisition software SPECTOR is written in C/C++ and handles all communication with the data acquisition hardware and instrumentation on the beam line. This software starts the experimental measurement and handles data visualization and storage. EPICS commands provided by the EPICS server were integrated into the acquisition flow of SPECTOR in the following sequence:

1. Researcher presses the “Start” button
2. Initialize electrometer
3. Start charge measurement
4. Start data acquisition from detectors
5. Open chamber valve
6. Current and charge values are read with detector data and displayed to the user
7. Chamber valve is automatically closed when a user preset charge value is reached
8. Final data from detectors and electrometer is read into the program

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV005

9. Data with the total charge is saved with the histogram spectrum

Figure 2 depicts how the SPECTOR user interface was modified to display the information from the electrometer. The whole implementation does not change the acquisition process for the researcher and can be enabled or disabled by selecting a menu option.

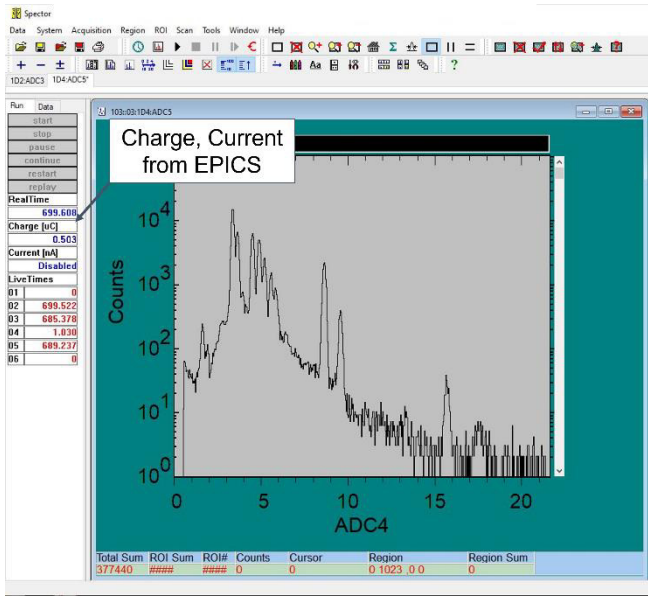


Figure 2: SPECTOR Integration.

RESULTS AND DISCUSSION

The implementation was very quickly accepted by researchers due to minimal changes introduced to the measurement procedure. It increased the number of samples that could be analyzed per beam time and proved to be more reliable when compared to the manual setup. An added benefit was that it stimulated researchers to suggest further automation on other beam lines which will further streamline the operating procedure in the laboratory. However, complications and challengers are also experienced during the implementation of the system. Communication delays and indeterministic response times from the IOCs running on the EPICS server created problem during the integration into SPECTOR as data was not always available when requested. This delay along with the mechanical delay of the pneumatic valve required modifications and redundancies to be added into the program flow of SPECTOR. SPECTOR was developed in cooperation with the IAEA [5] in 2002 and it mostly written in C with basic WIN32 libraries make it very difficult to modify due to unstructured code.

Further automation can be achieved by integrating the sample positioner located in the experimental chamber into EPICS. This would allow the system to automatically measure all the loaded samples with minimal supervision resulting in a further increase in laboratory efficiency. However, this would require precise sample position information in a feedback loop to confirm that the sample is located in the beam spot as not all samples are of standard shape and size. The integration of the acquisition system into EPICS would decrease the development time required to introduce further automation into the laboratory.

CONCLUSION

Expanding on the feasibility study/preliminary implementation of EPICS [1] at the Ruđer Bošković Institute Laboratory for Ion Beam Interactions, the advantages of such a control system were successfully demonstrated by adding automation to beam line measurements, and with it increased beam time utilization. The implementation of this simple system demonstrated the benefits of EPICS to the research staff and opened the doors to future automation.

This work implements only a small portion of the EPICS framework and future work is planned to incorporate other EPICS modules to expand the system. At the moment, the backward compatibility with the existing control system is limiting the implementation. Switching to an EPICS only control system will allow for further automation to both the accelerator and experimental systems.

REFERENCES

[1] D. Cosic, J. Radić, and M. Russo, "EPICS Implementation for Active Magnetic Field Control," in *SoftCOM*, 2020, pp. 1–3.

[2] K. Ishii, "PIXE and Its Applications to Elemental Analysis," *Quantum Beam Sci.*, vol. 3, no. 2, p. 12, 2019, doi: 10.3390/qubs3020012.

[3] D. Cosic, M. Bogovac, and M. Jakšić, "Data acquisition and control system for an evolving nuclear microprobe," *Nucl. Instruments Methods Phys. Res. Sect. B Beam Interact. with Mater. Atoms*, vol. 451, pp. 122–126, 2019, doi: 10.1016/j.nimb.2019.05.047.

[4] B. Franksen, "State Notation Language and Sequencer." https://www-csr.bessy.de/control/SoftDist/sequencer/ (accessed Oct. 04, 2021).

[5] M. J. Usher, "Data acquisition for engineers," *Microprocess. Microsyst.*, vol. 7, no. 2, pp. 84–85, 2002, doi: 10.1016/0141-9331(83)90277-6.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV006

AUTOMATED OPERATION OF ITER USING BEHAVIOR TREE SEMANTICS

W. Van Herck†, B. Bauvir, G. Ferro, ITER Organization, St. Paul lez Durance, France

Abstract

The inherent complexity of the ITER machine and the diversity of the ways it will be operated in different phases, like commissioning or engineering operation, poses a great challenge for striking the right balance between operability, integration and automation.

To facilitate the creation and execution of operational procedures in a robust and repeatable way, a software framework was designed and developed: the Sequencer. As a supporting framework for tasks that are mostly goal-oriented, the Sequencer's semantics are based on a behavior tree model that also supports concurrent flows of execution [1].

In view of its intended use in very diverse situations, from small scale tests to full integrated operation, the architecture was designed to be composable and extensible from the start. User interactions with the Sequencer are fully decoupled and can be linked through dependency injection.

The Sequencer library is currently feature-complete and comes with a command line interface for the encapsulation of procedures as system daemons or simple interactive use. It is highly maintainable due to its small and low complexity code base and dependencies to third party libraries are properly encapsulated.

Forecasted activities for this year include its use for the commissioning of plant systems, its incorporation as a foundation for ITER CODAC central monitoring and automation functions and the development of a graphical user interface.

INTRODUCTION

During the different phases of the ITER machine, many operational procedures will need to be defined, verified and executed in a traceable and maintainable way.

During regular operation, these procedures are mainly associated with Operational Tasks, e.g. venting, baking, etc. These procedures will very likely need to evolve in the lifetime of ITER, either by:

- changing the content and order of the different steps,
- changing parameters that influence the execution of the steps, or by
- increasing the automation of the execution by removing unnecessary user interactions as the procedure becomes more mature and trusted.

During the testing, calibration and commissioning of plant systems, local procedures will be defined to carry out tasks in a repeatable and traceable way. These procedures will likely be even more often adapted.

The Sequencer is a software tool meant to facilitate the creation, adaptation, approval and execution of those procedures. It also defines the format of these procedures, allowing version control and easy traceability. It provides a means to replace procedural documents with instructions and integrate automatic verification.

It will rely on the configuration, monitoring and control functions of the Supervision and Automation System (SUP) to carry out certain actions defined in the procedure. The Sequencer will thus need to adhere to the protocols defined by SUP.

This tool also has to provide the flexibility to be deployed and used in different environments:

- SUP interfacing with the Sequencer to execute a procedure,
- local activities using a standalone GUI,
- creation and editing can be done in an offline environment.

This paper describes the design and implementation choices and provides an outlook to future enhancements or extensions of the framework.

DESIGN

The language of operational procedures, whether for engineering operations or commissioning tasks, formulates certain goals that need to be achieved at each step. The Sequencer framework was based on behavior tree semantics since, as a formal language, it is generally better adapted to express such goal-oriented procedures than for example finite state machines. Goals can be described by a variety of rules that apply to sub-goals, such as: a Sequence succeeds if all its sub-goals succeed. Another advantage of behavior tree semantics is that it can express parallelism in a natural way. This is often required in operational procedures, where certain goals need to be achieved while maintaining conditions on the machine.

A procedure in the Sequencer library consists of tree structures of instruction objects and a workspace, providing access to variables that need to be shared or communicated between instructions. A procedure is executed by sending 'Execute' commands to a root instruction in the tree until it indicates failure or success. The root instruction is responsible for propagating this 'Execute' command further down the tree.

Figure 1 shows a simplified example of how the goal of starting up a plant system consists of either successfully activating the system (left branch) or, if that fails, executing steps to recover the system to a known and safe state (right branch). Each of those goals can in turn be expressed as a number of actions that need to be carried out sequentially.

† Walter.Vanherck@iter.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV006

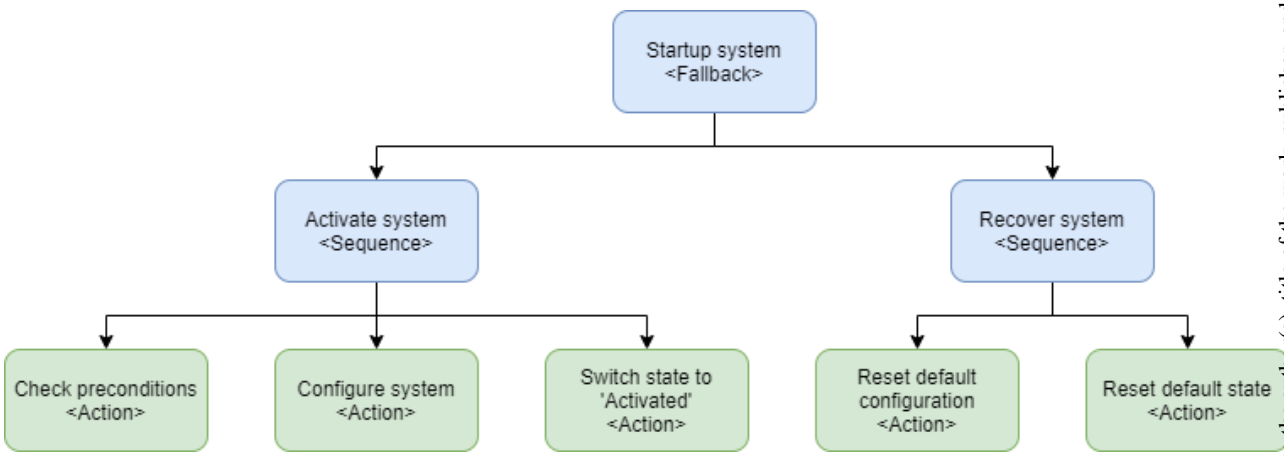


Figure 1: Example procedure for starting up a plant system.

The 'Action' instructions typically contain the domain specific commands or goals, while the compound instructions, such as 'Sequence' or 'Fallback', are domain agnostic.

The core library of the Sequencer framework contains generic instructions and variable types. Domain specific behavior or general extensions can be provided by a plugin mechanism, where extra instruction or variable types can be defined. By loading different plugins, the Sequencer framework can be used in different environments.

The framework also supports serialization of procedures in a human readable format (currently only XML is supported) to allow for version control and traceability.

IMPLEMENTATION AND QUALITY ASSURANCE

To support usage of the Sequencer framework for varied operational tasks and in different environments, composability and extensibility are key. Composability is provided by the basic building blocks of behavior trees: actions/goals are aggregated into compound goals. The Sequencer framework also allows to include a whole instruction tree from either the same or another procedure file. This allows designers of operational procedures to build complex procedures from ever simpler building blocks.

Extensibility is provided by a plugin mechanism and a generic API for instructions and variables. Behavior can then be customized by implementing instructions and variables that comply with this API and then exposing them as a plugin library.

The framework is implemented in C++11 and is highly maintainable due to its small codebase (currently ca. 5k lines of code) and the amount of decoupling between different components. The source code passes the current criteria for CODAC Software Integrity Level 1 (>95% unit test coverage, no major/critical/blocker issues).

User interaction with procedure execution, e.g. to allow for step-by-step execution or get visual feedback on the progress, is provided by a pure interface and concrete implementations of this user interface are injected into the framework (see Fig. 2). The core framework currently provides two such implementations:

- sequencer-cli: a command line interface to run procedure files with configurable amount of verbosity,
- sequencer-daemon: a non-interactive executable targeted to be run as a daemon process.

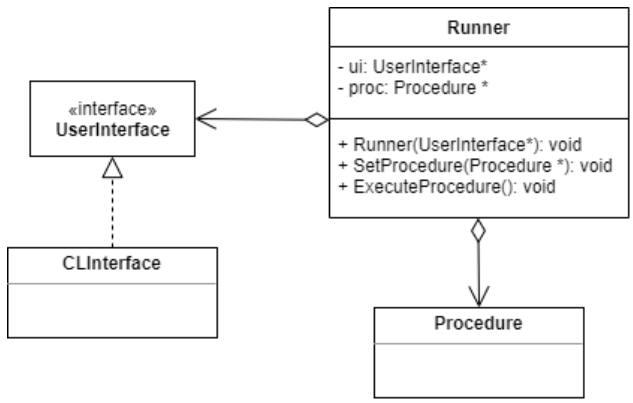


Figure 2: Constructor injection of UserInterface into the execution engine (Runner). Here, a command line interface (CLInterface) is shown as an example.

FORWARD LOOK

It is expected that the framework, but also its possible use cases, will continue to evolve due to changing and extending requirements. For reasons of compatibility and stability, such changes need to minimize the impact on the core library and should therefore be implemented in plugins whenever possible.

For the coming year, a number of use cases and further development activities have been identified:

- The framework will be used during the commissioning of plant systems: procedures will be developed to commission and automate routine tasks, such as starting up or shutting down the plant system. Concretely, it will be used for the commissioning of the Reactive Power Compensation and Harmonic Filter system and for the Site Acceptance Test and commissioning of the Magnetics Diagnostics.
- The Sequencer will also be evaluated for use in monitoring the ITER plant and automating central control

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV006

system activities. As part of these activities, a plugin is being developed for SUP specific instructions (e.g. for plant system configuration and verification).

- Lastly, a graphical user interface (GUI) is being developed that will enable users to easily create and edit Sequencer procedures. This GUI will also allow to interactively execute such procedures in multiple ways: step-by-step execution, breakpoints at given instructions, etc.

REFERENCES

[1] R. G. Dromey, "Formalizing the Transition from Requirements to Design", in *Mathematical Frameworks for Component Software Models for Analysis and Synthesis*, Nov. 2006, pp. 173-205. doi:10.1142/9789812772831_0006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV007

MACHINE LEARNING PROJECTS AT THE 1.5–GEV SYNCHROTON LIGHT SOURCE DELTA

D. Schirmer*, A. Althaus, S. Hüser, S. Khan, T. Schüngel
Center for Synchrotron Radiation (DELTA), TU Dortmund University, Germany

Abstract

In recent years, several machine learning (ML) based projects have been developed to support automated monitoring and operation of the DELTA electron storage ring facility. This includes self-regulating global and local orbit correction of the stored electron beam, betatron tune feedback as well as electron transfer rate (injection) optimization. Furthermore, the implementation for a ML-based chromaticity control is currently prepared. Some of these processes were initially simulated and then successfully transferred to real machine operation. This report provides an overview of the current status of these projects.

INTRODUCTION

DELTA is a 1.5–GeV electron storage ring facility operated by the TU Dortmund University as a synchroton light source [1] and as a facility for ultrashort pulses in the VUV and THz regime [2, 3]. Due to thermal orbit movements and magnetic field changes caused by different insertion device setups, the beam orbit and the betatron tunes may vary during storage ring operation. Therefore, autonomous local and global beam position corrections as well as self-adjusting tunes controls are important tasks, as otherwise sudden beam losses can occur. For this purpose, conventional, fully connected, shallow feed-forward neural networks (NNs) were investigated and have been successfully implemented. Both machine learning (ML) based controls were first simulated and tested on a detailed storage ring model within the Accelerator Toolbox (AT, [4, 5]) framework and were then successfully applied during real accelerator operation.

So far, the storage ring chromaticity values have been adjusted empirically based on experience. Setting of new values can only be done by time-consuming trial and error. For this reason, a ML-based algorithm for automated chromaticity adjustment is currently being prepared, very similar to the already implemented ML-based betatron tunes control. Classical, non-deep NNs are also used in this case.

At present, the electron transfer efficiency from the booster synchrotron to the storage ring is being optimized with the help of ML techniques, too. Here, NNs as well as Gaussian process regression (GPR) methods are explored.

ORBIT CORRECTION

Extensive studies for a ML-based orbit correction (OC) at the storage ring DELTA started already in 2017. Therefore, initially only the horizontal beam positions were disturbed by horizontally deflection corrector magnets (steerer) and the corresponding data pairs (orbit/steerer changes) were used as training data for supervised learning of fully connected neural NNs. First simulations have shown that already three-layered neural networks were able to learn correlations between beam position deviations and steerer strength changes. The application of such trained networks on the real storage ring results in similarly good beam position correction quality compared to conventional OC methods like SVD-based (singular value decomposition [6]) programs, however, with significantly fewer correction steps [7]. Subsequently, the ML algorithm was extended to both accelerator planes (x/y-coupled orbit), including weighted beam position monitor (BPM) signals. Thus, exposed positions in the DELTA storage ring (e.g., injection region, synchrotron radiation source points) can now be adequately considered in the ML-based OC.

In comparison with a more advanced numerical OC approach (qp-cone [8, 9]), the ML-based version results also in similar correction performance, which is scored by the weighted rms orbit error summed for both planes. On average the ML method still required fewer OC steps in this benchmark.

Exemplarity, some benchmark results are depicted in Fig. 1 (ML-based) and Fig. 2 (conventional). Even beam position deviations provoked by perturbations which were not applied during the training (e-j) could also be compensated equally. In all cases, after each provoked orbit disturbance, the residual weighted orbit error, as a measure for the OC

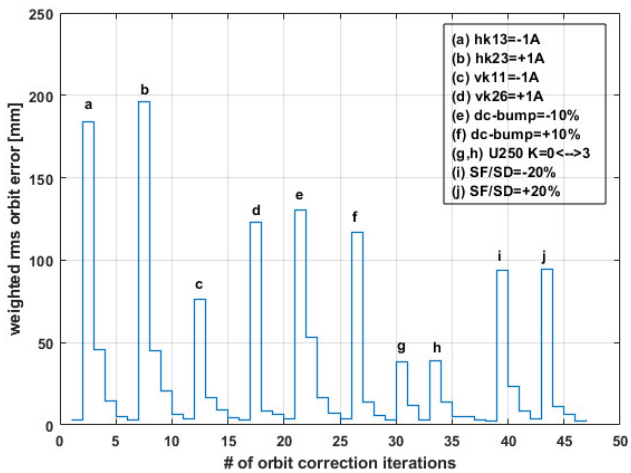


Figure 1: Individual correction steps for different scenarios of orbit deviations (a-j) performed with the ML-based OC program. In comparison to Fig. 2, similar final residual orbit qualities, scored by the weighted rms orbit error, are achieved in significantly fewer iterations.

* detlev.schirmer@tu-dortmund.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

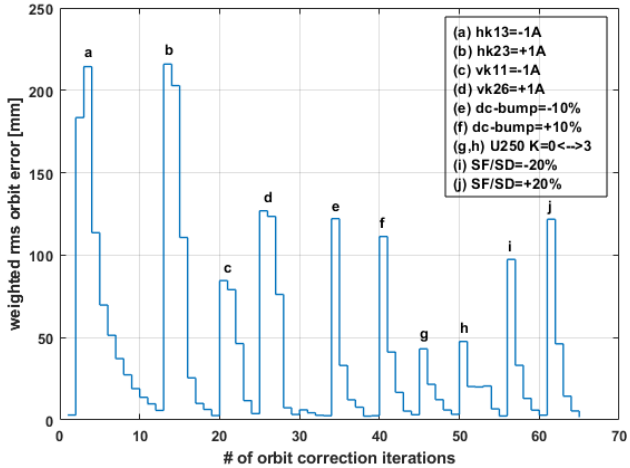


Figure 2: Number of orbit correction iterations to compensate the same orbit disturbing sources (a-j) as indicated in Fig. 1 performed with a qp-cone-based conventional orbit correction program [8, 9].

performance, converged to less than 3 mm. Further details of the ML-based orbit correction procedure are documented in [7, 10].

BETATRON TUNE CONTROL

To obtain appropriate data for NN training, the set values of seven independent quadrupole families located in the storage ring arcs were randomly changed and subsequently the associated tune shifts were recorded. Three-layered NNs were trained with experimental machine data as well as with simulated data based on a detailed lattice model of the storage ring [11]. Thereby, for supervised learning a variety of different gradient backpropagation methods were tested [12].

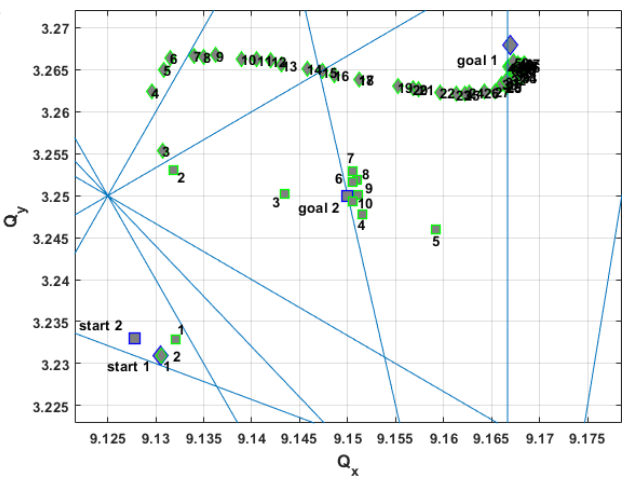


Figure 3: Validation of NNs trained with experimental data and applied to real machine operation. Two experiments demonstrate tune matching from start to goal tunes without beam losses. In the first experiment, 10 steps were performed (superconducting wiggler magnet switched on, rectangular markers) and in the second test 50 steps were executed (superconducting wiggler magnet switched off, diamond markers) [12].

With both data sources, comparable tune correction accuracies were achieved, both, in real machine operation and for the simulated storage ring model. In addition, it was also possible to perform tunes control 'crosswise', i.e., control of the real storage ring with NNs only trained by simulated model data; and vice versa, NNs only trained with real machine data and then applied to the simulation model. In all cases, the desired tune matching was successful.

In contrast to conventional PID control methods [13], trained NNs are able to approach the desired target tunes in fewer steps, which could enable a more controlled scanning in the tune diagram. Since the DELTA quadrupole power supplies lack the feature to drive synchronously in real-time, an iterative procedure for the NN-based tune control loop must necessarily be applied. Figure 3 depicts typical ML-based tune matching examples. Further examples and a more detailed description can be found in [12].

CHROMATICITY CONTROL

Similar to the ML-based tune control, the procedure can likewise be transferred to an automatic ML-based chromaticity control. At the DELTA storage ring, the chromaticity values are manually adjusted using 15 independent sextupole power supply circuits. However, to account for the symmetry of the optics, these circuits are grouped into 7 sextupole families during standard machine operation.

To acquire suitable ML training data, first, the sextupole strengths are scanned systematically for single and then randomly varied for all families. For each strength change, the associated chromaticity shifts are measured. Figure 4 visualizes corresponding simulation results for a DELTA storage ring model. With these data pairs (strength varia-

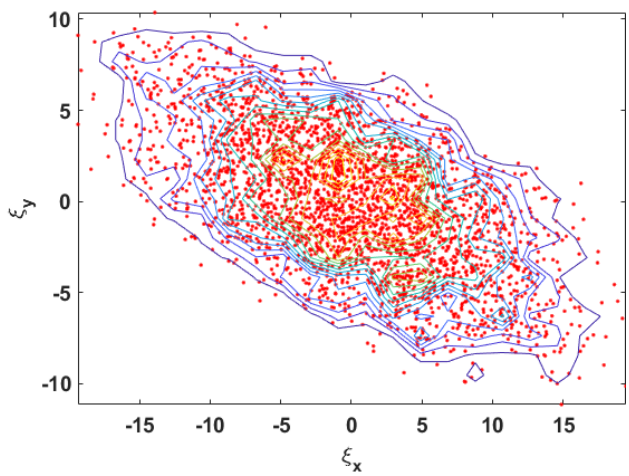


Figure 4: Distribution of 3000 chromaticity shifts invoked by uniformly randomized strength variations of seven independent sextupole families. The data are obtained by AT optics calculations based on a Delta storage ring model and are used for supervised training of NNs.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV007

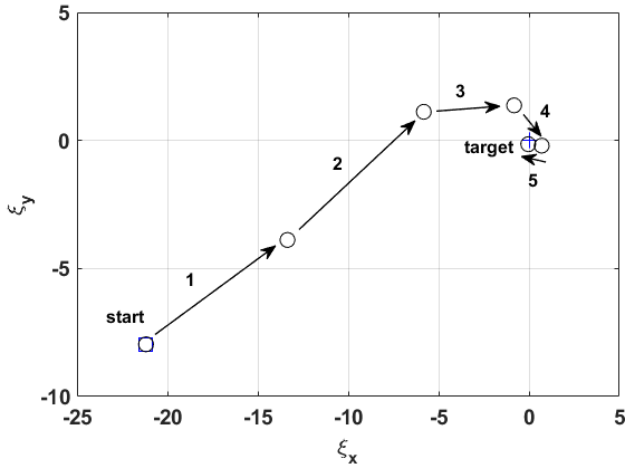


Figure 5: Example for verification of NNs trained by simulated data (see Fig. 4) and applied to the DELTA storage ring model. The desired target values for compensated chromaticity (goal: $\xi_x = \xi_y = 0$) were reached in iterative steps starting at the setting for natural chromaticity ($\xi_x = -21$, $\xi_y = -8$, sextupoles switched off).

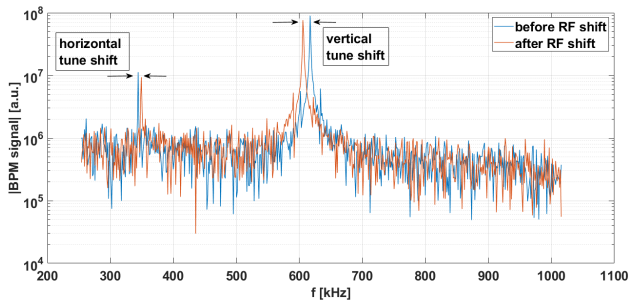


Figure 6: FFT beam spectrum from turn-by-turn orbit data recorded at a dedicated 'fast' BPM (blue) and after (red) a cavity radiofrequency (RF) variation of 5 kHz. The horizontal and vertical chromaticities are calculated by determining the betatron tune peak shifts induced by the cavity RF variation.

tions and chromaticity shifts), 'chromaticity-models' will be trained, which later could serve to adjust and control the chromaticity values automatically during real machine operation. Figure 5 illustrates an example for simulated chromaticity matching performed with a 3-layered NN which has been trained by conjugate gradient backpropagation using the data depicted in Fig. 4. The natural chromaticity ($\xi_x = -21$, $\xi_y = -8$) which occurs with all sextupole switched off (start) can be adjusted to full chromaticity compensated values ($\xi_x = \xi_y = 0$) by the ML-based control loop. The step size (number of iterations) is adjustable and depends mainly on the granularity of the trainings data.

For chromaticity determination in real ring operation, the cavity radiofrequency (RF) must be shifted and then the tune shifts ΔQ is determined via an FFT spectrum from turn-by-turn orbit data at a dedicated 'fast' BPM (see Fig. 6 as an example). With $\Delta Q = \xi \cdot \Delta p/p$ follows for the chromaticity $\xi = \Delta Q \cdot p/\Delta p = -\alpha_c h \Delta f_\beta / \Delta f_{\mathrm{RF}}$. Δf_{RF} corresponds to changes of the cavity radiofrequency, Δf_β is the measured betatron frequency shift, h is the harmonic number and the momentum compaction factor $\alpha_c = (\Delta L/L)/(\Delta p/p)$ relates the relative orbit path length change $\Delta L/L$ to the relative momentum change $\Delta p/p$. Work on this project has just started in the framework of a master's thesis.

INJECTION OPTIMIZATION

Already in 2005, first attempts were made to optimize the electron transfer rate (injection efficiency) from the booster synchrotron to the storage ring by a combination of genetic algorithms and neural networks [14, 15]. Currently, this idea is being resumed but now with an expanded number of parameters and with a significantly enlarged database for ML-based training. In addition to the strength settings of the transfer line magnets, the injection elements of the storage ring (e.g., kicker magnets, magnets of a static injection bump) as well as the trigger timings of all pulsed transfer line and injection magnets are now taken into account, too.

In total, currently up to 30 different parameters can be varied systematically or randomly. For each individual injection parameter variation, the corresponding change of the injection efficiency is measured simultaneously. In this way, several thousand pairs of experimental data sets were generated which served as inputs for various machine learning techniques. So far, classical feed-forward neural networks (NNs) and Gaussian process regression (GPR) methods were used for modeling [16, 17]. First training results are shown in Fig. 7 (GPR-based) and Fig. 8 (NN-based). They show the trained model predictions of the injection efficiencies as a function of approx. 750 measured transfer efficiencies.

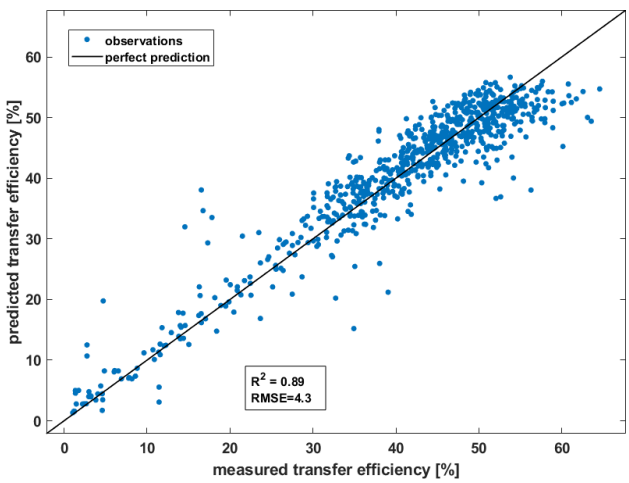


Figure 7: Injection efficiencies predicted by a Gaussian process regression (GPR) model versus measured efficiencies. The machine learning was performed with approx. 750 measured data sets. The GPR applied the nonisotropic exponential kernel function for model adaption [16]. The fit results in a correlation coefficient of $R^2 = 0.89$ and a root mean squared value (RMSE) of 4.3.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

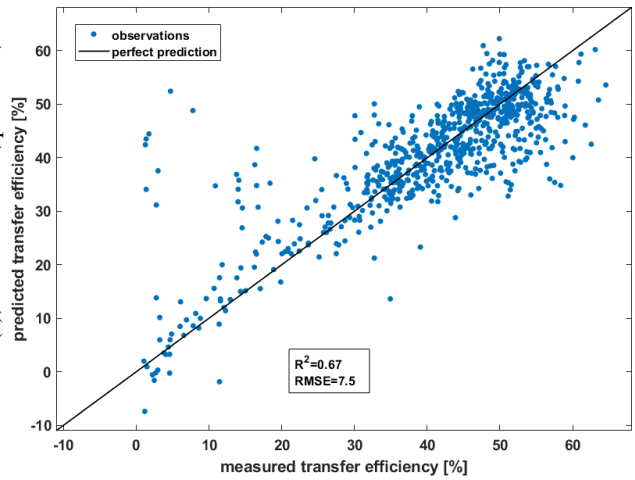


Figure 8: Injection efficiencies predicted by a model based on a narrow neural network (NN) versus measured efficiencies. The machine learning was performed with the same 750 data sets as used for GPR-based modelling. The NN is composed of three fully connected ReLU-layers (28/10/1) [16]. The fit results in a correlation coefficient of $R^2 = 0.67$ and a root mean squared value (RMSE) of 7.5.

The plots were obtained applying the k-fold cross validation method. Therefore, the training data sets were randomly shuffled and then divided into k partitions. For each training-validation iteration a different partition for validation was used. The remaining data was applied for testing. Thus, each data partition was used once for validation and $k - 1$ times for training.

Both ML methods find clear correlations, whereby the GPR-trained model results in a significantly larger correlation coefficient R and thus seemed to be more suitable for transfer rate modeling. The next step is to apply these models to keep the injection efficiency at a high level in an automated way using an optimization algorithm (e.g. BFGS [18] or Bayesian optimization [19]) without the need for operator interventions [20]. These studies are currently being continued.

SUMMARY

ML-based techniques are increasingly replacing conventional optimization methods in the fields of particle accelerator controls. At the electron storage ring facility DELTA, it was demonstrated that specially designed and trained neural networks are suitable for a variety of control tasks. Supervised training of the NNs was performed with data recorded during real machine operation and with simulation data based on accelerator models. Compared to classical numerical methods, it was shown that the different ML-based applications could achieve comparable correction accuracies. In contrast to conventional numerical methods, neural networks are in principle able to converge to the desired target values in fewer iterations, resulting in a faster correction convergence behaviour. Inspired by a variety of ML-supported applications in other scientific fields, additional use cases in the domain of accelerator controls are being evaluated. First ideas for a neuro-fuzzy feedback loop, e.g., for the water cooling system of the new EU-type RF cavity installed at the DELTA storage ring, have already been discussed [7]. It is planned to migrate all ML-based programs to a dedicated GPU-based ML server which provides a powerful hardware platform and a more user friendly ML framework (ML workflow) for future, more sophisticated ML applications.

ACKNOWLEDGEMENT

We thank all colleagues of the DELTA team for many inspiring advises and suggestions as well as for providing sufficient data acquisition time during accelerator shifts.

REFERENCES

[1] M. Tolan, T. Weis, C. Westphal, and K. Wille, "DELTA: Synchrotron light in nordrhein-westfalen", *Synchrotron Radiation News*, Vol. 16, pp. 9–11, Mar. 2003. doi:10.1080/08940880308603005

[2] S. Khan *et al.*, "Coherent Harmonic Generation at DELTA: A New Facility for Ultrashort Pulses in the VUV and THz Regime", in *Synchrotron Radiation News 24*, Vol. 18 (2011). doi:10.1080/08940886.2011.618092

[3] S. Khan *et al.*, "Generation of Ultrashort and Coherent Synchrotron Radiation Pulses at DELTA", *Synchrotron Radiation News*, Vol. 26, pp. 25–29, May 2013. doi1:0.1080/08940886.2013.791213

[4] Accelerator Toolbox (AT), http://atcollab.sourceforge.net/index.html.

[5] A. Terebilo, "Accelerator Modeling with Matlab Accelerator Toolbox", in *Proc. of 19th Particle Accelerator Conference (PAC'01)*, Chicago, USA, 2001, pp. 3203–3205. doi:10.1109/PAC.2001.988056

[6] M. Grewe, "SVD-basierte Orbitkorrektur am Speicherring Delta", dissertation, TU Dortmund, Germany, 2005.

[7] D. Schirmer, "Intelligent Controls for the Electron Storage Ring DELTA", in *Proc. 9th Int. Particle Accelerator Conf. (IPAC'18)*, Vancouver, Canada, 2018, pp. 4855–4858. doi:10.18429/JACoW-IPAC2018-THPML085

[8] S. Kötter, B. Riemann, and T. Weis, "Status of the Development of a BE-Model-Based Program for Orbit Correction at the Electron Storage Ring DELTA", in *Proc. 8th Int. Particle Accelerator Conf. (IPAC'17)*, Copenhagen, Denmark, 2017, pp. 673–675. doi:10.18429/JACoW-IPAC2017-MOPIK065

[9] S. Kötter, A. Glassl, B. D. Isbarn, D. Rohde, M. Sommer, and T. Weis, "Evaluation of an Interior Point Method Specialized in Solving Constrained Convex Optimization Problems for Orbit Correction at the Electron Storage Ring at DELTA", in *Proc. of 9th Int. Particle Accelerator Conf. (IPAC'18)*, Vancouver, Canada, 2018, pp. 3507–3510. doi:10.18429/JACoW-IPAC2018-THPAK114

[10] D. Schirmer, "Orbit Correction with Machine Learning Techniques at the Synchrotron Light Source DELTA", in *Proc. of 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV007

New York, USA, 2019, pp. 1426–1430. doi:10.18429/JACoW-ICALEPCS2019-WEPHA138

[11] D. Schirmer and A. Althaus, "Integration of a Model Server into the Control System of the Synchrotron Light Source DELTA", in *Proc. of 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, USA, Oct. 2019, pp. 1421–1425. doi:10.18429/JACoW-ICALEPCS2019-WEPHA137

[12] D. Schirmer, "Machine learning applied to automated tunes control at the 1.5 GeV synchrotron light source DELTA", in *Proc. of 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, SP, Brazil, May 2021, pp. 3379–3382. doi:10.18429/JACoW-IPAC2021-WEPAB303

[13] P. Hartmann, J. Fürsch, R. Wagner, T. Weis, and K. Wille, "Kicker Based Tune Measurement for DELTA", in *Proc. of 8th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC'07)*, Venice, Italy, May 2007, pp. 277–279.

[14] T. Büning, D. Müller, "Entwurf und Vergleich unterstützender Systeme zur Verbesserung der Injektionseffizienz von DELTA, basierend auf Evolutionsstrategien und neuronalen Netzen", diploma thesis, Computer Science Dept., TU Dortmund University, Germany, 2005.

[15] D. Schirmer *et al.*, "Electron Transport Line Optimization Using Neural Networks and Genetic Algorithms", in *Proc. of 10th European Particle Accelerator Conference (EPAC'06)*, Edinburgh, Scotland, 2006, pp. 1948–1950.

[16] MATLAB/SIMULINK and Neural Network, Fuzzy Logic, Genetic Algorithm Toolboxes, Release 2017b, The MathWorks, Inc., Natick, Massachusetts, United States.

[17] C. E. Rasmussen and C. Williams, "Gaussian processes for machine learning", MIT Press, Cambridge, Mass., ISBN 9780262182539, 2005.

[18] Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, https://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm

[19] J. H. Snoek, H. Larochelle, R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms", https://arxiv.org/abs/1206.2944, 2012.

[20] Chenran Xu, "Baysesian optimization of Injection Efficiency at KARA using Gaussian Processes", Master's Thesis, Karlsruhe Institute of Technology, Germany, 2020.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV008

ONLINE AUTOMATIC PERFORMANCE OPTIMIZATION FOR THE ELETTRA SYNCHROTRON

G. Gaio†, S. Krecic, F. Tripaldi, Elettra-Sincrotrone Trieste S.C.p.A., Trieste, Italy

Abstract

Online automatic performance optimization is a common practice in particle accelerators. Beside the attempts based on Machine Learning, which is effective especially on non-linear systems and images but are very complex to tune and manage, one of the most simple and robust algorithms, the simplex Nelder Mead, is extensively used at Elettra to automatically optimize the synchrotron parameters. It is currently applied to optimize the efficiency of the booster injector by tuning the pre-injector energy, the trajectory and optics of the transfer lines, and the injection system of the storage ring. It has also been applied to maximize the intensity of the photon beam on a beamline by changing the electron beam position and angle inside the undulator. The optimization algorithm has been embedded in a Tango device that also implements generic and configurable multi-input multi-output feedback systems. This optimization tool is usually included in a high-level automation framework based on Behavior Trees [1] in charge of the whole process of machine preparation for the experiments.

INTRODUCTION

Similarly to most of the modern light sources, the Elettra 2-2.4GeV synchrotron relies on feedback systems to keep the beams stable and, more recently, on automatic optimization systems to help operators in maximizing the performance.

Feedback and optimization systems look in some aspects very similar. They have both sensors, actuators and have in common the objective of minimizing the distance between sensors and a reference. In feedback systems the reference is a setpoint of a machine parameter that have to kept fixed no matter the noise or external perturbations affecting the accelerator. In optimization problems the reference is still present but usually set to an arbitrarily very high /very low value if the intention is to maximize / minimize the sensor value.

By the way, excluding the algorithmic part, the software internal components (acquisition, command and control) are the same.

In light sources programmable feedback and/or optimization tools are quite common. Some of them are implemented as high-level applications [2,3] others are server applications [4].

At Elettra a programmable C++ Tango server, called MIMOFB (Multi Input Multi Output Feedback), has been developed to replace legacy applications implementing slow feedback systems and to implement automatic optimization procedures.

† email: giulio.gaio@elettra.eu

MIMOFB

When configured as a feedback, the MIMOFB implements a correction scheme based on a linear model that link sensors and actuators. This relationship is an approximation that is empirically calculated by measuring the perturbation generated on the sensors by one actuator at a time. The result of this process is a matrix, formally a Response Matrix (RM). The product between the inverse of the response matrix, usually inverted using the Singular Value Decomposition (SVD) algorithm, and the error vector returns the values to subtract from actuators to minimize the distance between the sensors and the reference.

Regarding optimization, the MIMOFB implements only model-less optimization schemes. In this case the objective function F (1) is the sum of the normalized distances between N sensor values and corresponding references multiplied by the sensor weights.

$$F = \sum_{i=0}^{N} w_i * \frac{abs(x_i - x_i^{ref})}{x_i^{\max_thres} - x_i^{\min_thres}} \quad (1)$$

The configuration of a MIMOFB device is stored in several device properties.

For each sensor the developer has to configure:

- the sensors Tango attributes
- the number of sensor readings per cycle
- descriptive label
- weight
- minimum threshold value
- maximum threshold value
- dead-band
- sensor update rate in ms
- type of filtering (none, mean, median) when the number of readings per cycle is higher than one
- Tango device allowed states
- list of actuators affecting the sensors

For each actuator it is required to specify:

- the actuator Tango attribute
- descriptive label
- minimum threshold value
- maximum threshold value
- scan range
- RM kick; this value is used in response matrix calculation and as the initial step in optimization algorithms
- maximum backlash (useful when working with motors)
- dead-band
- maximum difference between last read and set value
- RM kick settling time (msec.); this value changes proportionally to RM kick amplitude

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV008

- constant settling time in msec.
- tango device allowed states.

Thus, the repetition frequency of the algorithm depends only by the refresh rate and the settling time of the I/O. Acquiring sensors and setting actuators could be performed in series or parallel. When configured in series one iteration lasts the sum of all refresh and settling times. When configured in parallel, the repetition period is equal to the sum of the slowest sensor and actuator response. The repetition frequency can be tuned further by acting on a time factor that compress or expand all settling times.

Feedback Algorithm

The response matrix calculation is performed by the MIMOFB. At the end of the procedure the user can choose to load the new response matrix or just save it. In the response matrix inversion, the SVD singular values can be weighted and weights can be saved.
When the feedback is running, users can activate an adaptive procedure that improve the matching between the response matrix and the real machine. Basically, after a given feedback iteration, one actuator at a time performs a kick. The corresponding perturbation is acquired by the sensors to update the corresponding part of the response matrix that will be afterwards inverted for the next feedback step.

A procedure assists the user to estimate what is the optimal kick amplitude used for the response matrix calculation and at the start of the optimization procedures. The optimal kick amplitude is a tradeoff between a step that increases the *rms* of the sensor readings by at least 50% and the minimum step that drive at least one of the sensors out of range.

The closed loop algorithm is a proportional-integral-derivative controller (PID). The user can switch between the PID default configuration and a predefined one that the closed loop gain and guarantee a better feedback stability. Users can enable the anti-windup, the dead-band control and the slew-rate control on the actuators.

Optimization Algorithms

At the moment there are four model-less optimization algorithms available:

- 1D scan, one parameter
- 2D scan, two parameters
- Simultaneous Perturbation Stochastic Approximation (SPSA) [5], multi-parameter
- Nelder Mead [6], multi-parameter.

In the near future many model free algorithms provided by the NLOpt library [7] will be integrated in the server.

The optimization algorithm could run one shot or, by means of a programmable optimization scheduler (see Fig. 1), cycle between different combinations of optimization algorithms and actuators. For NM and SPSA the user can limit the maximum number of iterations or seconds per optimization. During execution, if no improvement occurs within one third of the remaining time, the optimization stops.

Monitoring and Recovery

The MIMOFB stops in fault state when sensor/actuator values and their statuses are out of allowed values for a number of consecutive cycles. Afterwards, the feedback/optimization can automatically restart once sensor and actuators are back in range.

When enabled, the processing automatically disables the sensors whose value remain constant for more than two samples or equal to zero.

If the sensor value is an average of more then one sample (mean/median), a threshold fixes the maximum number of outliers before marking the sensor invalid.

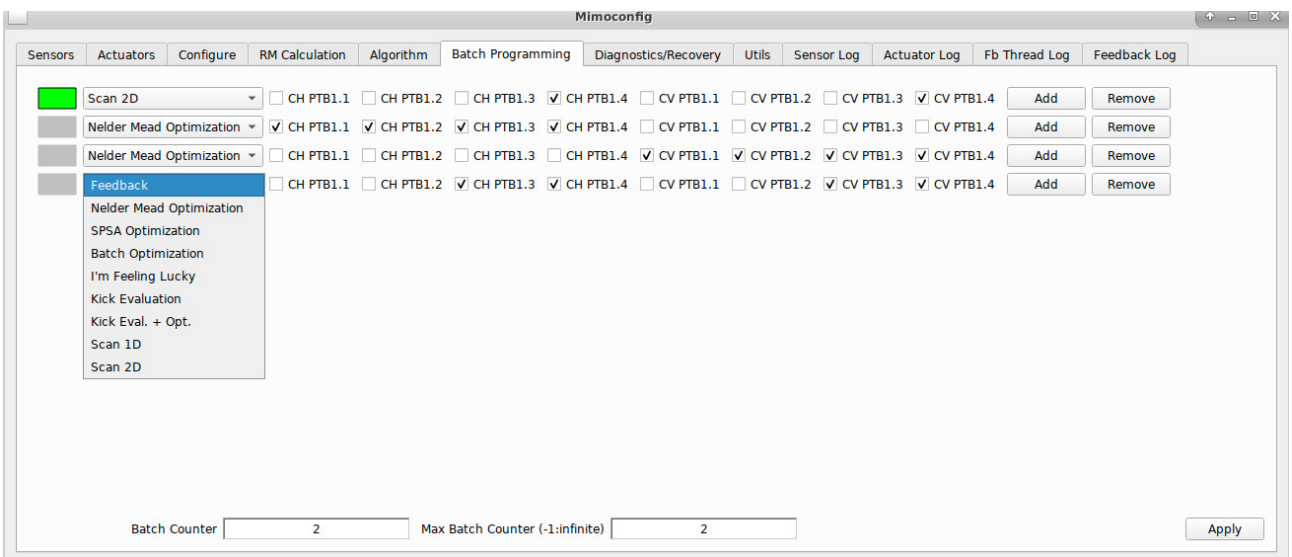


Figure 1: Configuration panel of the MIMOFB. The "Batch Programming" tab configures the optimization scheduler (example). CHV_PTBX.X are corrector magnets of a transfer-line. The optimization target is the booster accumulated current. The "I'm Feeling lucky" optimization mode couples randomly actuators and optimization algorithms.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV008

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Sensor and actuator data and error messages are stored into dedicated buffers (see Fig. 2). Similarly, the optimization / feedback engine has a dedicated logging buffer for saving error and debugging messages.

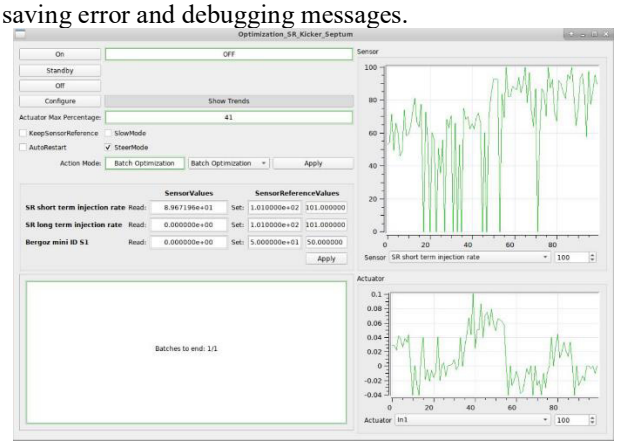


Figure 2: MIMOFB main control panel with logging buffers.

Sequencers Integration

The MIMOFB can be configured to execute a sequence of instructions, based on sequencers [1], at the start and at the end of the correction / optimization process.

Each sensor reading can be preceded and followed by the execution of a sequencer. Similarly, a sequencer can be executed before and after the setting of an actuator. In all cases the correction/optimization process waits sequencers to complete their task before going on. If the sequencer will end its task in fault state then the MIMOFB itself will fail.

The sequencers greatly expand the capabilities of the MIMOFB. A significant example is the alignment procedure, based on MIMOFB, of an electron/laser beam based on intercepting fluorescent screens. In this case the acquisition of the position of the beam on the screen has to be preceded by the insertion of the screen and followed by its extraction (to let the beam go to the next screen). This mechanical process is demanded to pre and post sensor sequencers that can be easily integrated without touching any internal logic of the MIMOFB. In another example a pre-sensor sequencer selects the most reliable sensor between two by acting on the weights of the sensors.

OPTIMIZATIONS

At Elettra the MIMOFB has slowly replaced all legacy high-level applications used for machine tuning and beam stabilization. MIMOFB implements slow local and global orbit correction in the storage ring, tune feedback, RF path length feedback and booster-to-storage ring transfer-line orbit feedback. However, the most interesting applications of the MIMOFB concern the optimization that, until recently, was performed manually by operators.

In Elettra the machine optimizations are one of the most critical steps in the automatic process that recovers the machine from a faulty state and gives back light to the users.

The optimization process has to be:

- maintainable by the operators
- aware of the machine state
- resilient to external perturbations
- able to restore the machine in case of any failure.

Basically, in all optimization schemes the Nelder-Mead (NM) algorithm together with the programmable optimization scheduler gets the best results. At Elettra the optimizer is used routinely for tuning ([description];[actuators];[algorithm];[target function]):

- pre-injector energy (see Fig. 3, num. 1); one actuator, pre-injector HV power supply; algorithm scan-1D; booster accumulated current
- pre-injector to booster RF phase matching (see Fig. 3, num. 2); one actuator, rf phase shifter; algorithm scan-1D; booster accumulated current
- booster to storage ring transfer line orbit (see Fig. 3, num. 5); four actuators, corrector magnet power supplies; algorithm Nelder-Mead; injection efficiency (see Fig. 3)
- storage ring injection system (Fig. 3, num. 6); four actuators, HV power supplies; algorithm Nelder-Mead; injection efficiency.

The optimizer is also used for tuning during machine preparation or machine physics shifts:

- pre-injector to booster transfer-line orbit (see Fig. 3, num. 3); four actuators, corrector magnet power supplies; algorithm Nelder-Mead; booster accumulated current
- pre-injector to booster transfer-line optics (see Fig. 3, num. 4); two actuators, quadrupole magnet power supplies; algorithm Nelder-Mead; booster accumulated current
- beamline photon flux (see Fig.3, num. 7); four actuators, position/angle at source point (eight correctors); algorithm Nelder Mead; beamline photodiode (see Fig. 4)

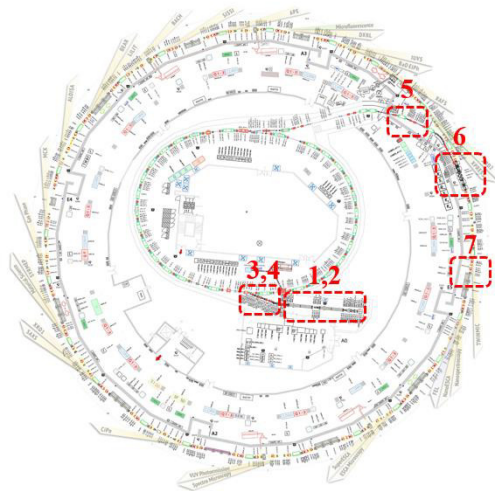


Figure 3: Elettra automatic optimized systems

CONCLUSION

The MIMOFB Tango server has become a key component in the Elettra operations. Thanks to the fact that it is a server, it can be embedded in the high level software framework based on sequencers, which carries out most of the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV008

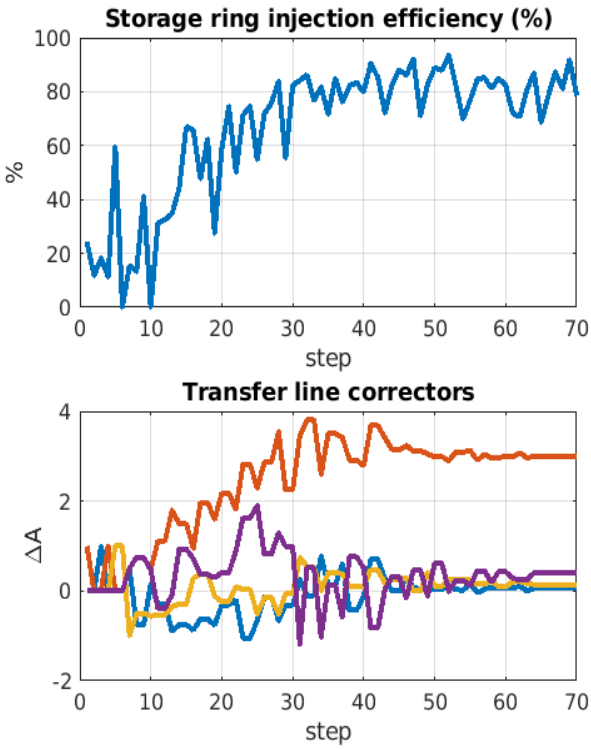


Figure 4: Optimization of the Elettra injection efficiency by changing the last correctors of the booster to storage-ring transfer line

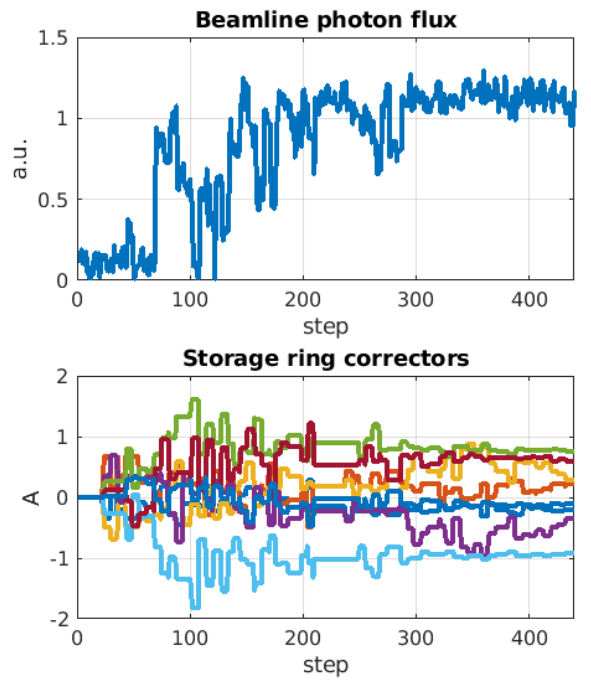


Figure 5: Optimization of the photon beam flux on a beamline by changing position and angle of the electron beam inside the undulator. Eight correctors are involved in the local orbit optimization procedure [8]. The batch programming engine runs the optimization for two cycles. In the second cycle the initial step amplitude was half the one of the first cycle.

tasks that operators should perform manually to prepare the machine and the electron beam for the beamline experiments.

Since the optimization procedures have to run without any human supervision, we have preferred robust and model-less algorithms with a minimum number of hyperparameters to tune. For the same reason we chose to limit to four the number of parameters involved contemporary in any unsupervised optimization process.

Programming batches can execute several combinations of optimization algorithms and different actuators to expand the limits of using just one setup. The Nelder-Mead restart technique [9] that can be easily implemented with batch programming speed up the process of finding the best working point. The necessity for any online algorithm to have a minimum signal for starting, can be mitigated with the possibility to performing 1D-2D wide range scans for signal recovering before running more sophisticated algorithms.

REFERENCES

[1] G. Gaio, P. Cinquegrana *et al.*, "A framework for high level machine automation based on Behavior Trees", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCSC'21), Oct 2021, paper WEAL02, this conference

[2] E. Piselli and A. Akroh, "New CERN Proton Synchrotron Beam Optimization Tool", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 692-696. doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

[3] S. Tomin, I. Agapov, *et al.*, "Online optimization of European XFEL with Ocelot", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1038-1042. doi:10.18429/JACoW-ICALEPCS2017-WEAPL07

[4] P. Bell, M. Sjöström, *et al.*, "A General Multiple-Input Multiple-Output Feedback Device in Tango for the MAX IV Accelerators", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Oct. 2019, pp. 1084–1088. doi:10.18429/JACoW-ICALEPCS2019-WEPHA012

[5] J. Nelder and R. Mead, "A Simplex Method for Function Minimization", *The Computer Journal*, vol. 7, Jan. 1965, pp. 308-313. doi:10.1093/COMJNL/7.4.308, corrections in doi:10.1093/comjnl/8.1.27

[6] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", *IEEE Transactions on Automatic Control*, vol. 37, pp. 332-341, March 1992. doi:10.1109/9.119632

[7] NLopt, https://nlopt.readthedocs.io/

[8] C. J. Bocchetta, D. Bulfone, *et al.*, "First operational results with the ELETTRA fast local feedback system", in *Proc. 17th IEEE Particle Accelerator Conference (PAC'97)*, Vancouver, Canada, May 1997, pp. 2356-2358. doi:10.1109/PAC.1997.751207

[9] Q. H. Zhao, D. Urosević, *et al.*, "A restarted and modified simplex search for unconstrained optimization", *Computers & Operations Research*, vol. 36, 2009, pp. 3263-3271. doi:10.1016/j.cor.2009.03.005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV010

R&D OF THE KEK LINAC ACCELERATOR TUNING USING MACHINE LEARNING

A. Hisano[†1], M. Iwasaki[#1,2,3,4] ,I. Satake[5], M. Satoh[5,6]
H. Nagahara[4,3], N. Takemura[4,3], Y. Nakashima[4,3], and T. Nakano[3,4]
[1]Osaka City University Graduate School of Science, Osaka, Japan
[2]Nambu Yoichiro Institute of Theoretical and Experimental Physics (NITEP), Osaka, Japan
[3]Research Center for Nuclear Physics, Osaka University (RCNP), Osaka, Japan
[4]Osaka University Institute for Datability Science (IDS), Osaka, Japan
[5]High Energy Accelerator Research Organization (KEK), Ibaraki, Japan
[6]The Graduate University for Advanced Studies (SOKENDAI), Department of Accelerator Science, Kanagawa, Japan

Abstract

We have developed a machine-learning-based operation tuning scheme for the KEK e-/e+ injector linac (Linac), to improve the injection efficiency.

The tuning scheme is based on the various accelerator operation data (control parameters, monitoring data and environmental data) of Linac.

For the studies, we use the accumulated Linac operation data from 2018 to 2021.In this paper, we show the results on our R&Ds of, 1. visualization of the accelerator parameters (~1000) based on the dimensionality reduction, and, 2. accelerator tuning using the deep neural network (DNN). In the latter R&D, estimation of the accelerator parameters using DNN, and the accelerator simulator based on the Generative Adversarial Network (GAN) have been studied.

INTRODUCTION

We have developed an operation tuning scheme for the KEK e-/e+ injector linac (Linac) [1]. The Linac accelerator is a 600m long injector linac to distribute electrons and positrons to four ring accelerators: the Photon Factory (PF), the PF Advanced Ring (PF-AR), the SuperKEKB Electron Ring (HER) and the Positron Ring (LER). The Linac accelerator is capable of operating at up to 50 Hz in two bunches (96 ns apart), which is equipped with 100 beam position monitors (BPM), 30 steering magnets and 60 RF monitors.

Though the precise beam tuning and high injection efficiency are required, there exist several problems on the accelerator tuning as: 1. A lot of parameters (~1000) should be tuned, and these parameters are intricately correlated with each other; and 2. Continuous environmental change, due to temperature change, ground motion, tidal force, etc., affects to the operation tuning.

To solve the above problems, we have developed, 1. visualization of the accelerator parameters (~1000) trend/correlation distribution based on the dimensionality reduction to two parameters, and 2. accelerator tuning using the deep neural network, which is continuously updated with the accelerator data to adapt for continuous environmental change.to adapt the environment changes.

In this paper, we show the results on our R&D. For the studies, we use the electron beam data for Super KEKB injection accumulated in 2018 Nov. - 2021 Jun. This beam data includes the following parameters.

- 500 operating parameters (Steering magnet)
- 732 environmental parameters

In addition, we use the ratio of the upstream (SP_A1_M) and downstream (SP_58_0) charges of the accelerator as a quantitative measure of the injection efficiency of the accelerator.

VISUALIZATION OF THE ACCELERATOR PARAMETERS

In this study, we use dimensionality reduction with a Variational AutoEncoder (VAE) [2] as a visualization method of accelerator parameters. VAE is a neural network consisting of two networks: the encoder, which converts the input data into "latent variables" of arbitrary dimensions, and the decoder, which reconstructs the input data from the latent variables. Here, VAE assumes that "latent variables" are normally distributed, so that latent variables for similar trend input data are closely distributed when they are plotted on a space. Using this property, we can visualize the accelerator parameters by dimensionally reducing them to two-dimensional "latent variables" and plotting them in a two-dimensional space.

To check the visualisation performance by VAE, we created an accelerator parameters dataset containing 1232 parameters (operating params + env params). Figure 1 shows the results of the visualisation of the accelerator parameters accumulated between 2018 Nov. and 2021 Jun. using VAE trained by 2018 Nov to 2021 May data. The output results are coloured according to the injection efficiency of the input accelerator data.

As a result of Figure. 1, the accelerator parameter dataset containing 1232 parameters is visualized by VAE with di-

† m20sa029@uv.osaka-cu.ac.jp
# masako@osaka-cu.ac.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV010

mensionality reduction to two dimensions. And, the visualisation results show that in the short term(~1month) the accelerator trend does not drastically change, but in the long term(>3month) accelerator trend vary over a wide range.

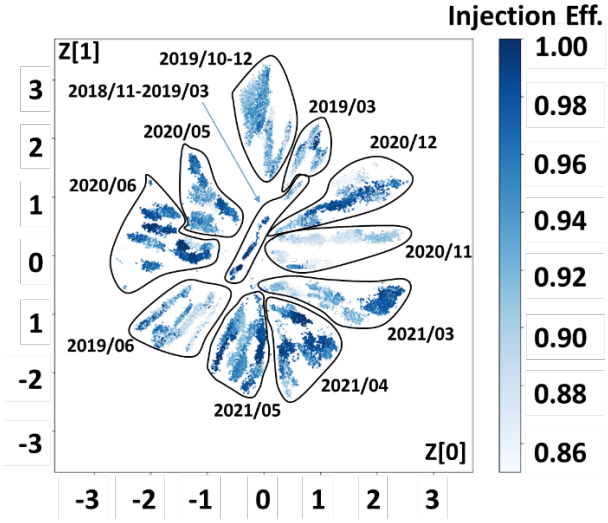


Figure 1: VAE output (two-dimensional Latent variables) from the 2018 Nov to 2021 June accelerator data. VAE is trained using 2018 Nov to 2021 May data. The output results are coloured according to the injection efficiency of the input accelerator data.

ACCELERATOR TUNING USING THE DEEP NEURAL NETWORK

The operating environment of the accelerator is continuously changing. In addition, the correlations between the parameters are complex. In this situation, to continuously adapt to the environmental changes to get the high injection efficiency, the environment driven DNN (Reinforcement ML) is a good candidate. However, to apply the Reinforcement ML to the operation tuning, we need 1. Estimation of the acc. parames using DNN, 2. realistic accelerator simulator. In the following, we introduce these R&D.

Estimation of the Acc. Parames using DNN

For the reinforcement learning, it is important to start from the optimized (the best) accelerator parameter value, so that ML doesn't need to search in wide parameter range.In addition, we need to obtain the relationship between efficiency and accelerator parameters, for the effective parameter optimization.

Therefore, we have developed a method to estimate an injection efficiency using DNN with accelerator parameters inputs, to assure to get the relationship between them.

In this study, we designed a Regression-DNN that takes 1232 accelerator parameters (operating params and env params) as input and outputs the injection efficiency. The implementation of the DNN is based on Tensorflow [3].

Evaluate the performance of DNN To evaluate the performance of DNN, we extract 150,000 events data from Linac data accumulated in 2018 Nov. - 2021 Jun. Note that the injection efficiency of the dataset is extracted to be uniformly distributed over 0.85. Out of 150,000 events, we use 135,000 events for training and 15,000 events for validation.

We trained and validated Regression-DNN on dataset described above. As a result, the mean squared error (MSE) between the predicted injection efficiency and the real injection efficiency is 0.00030 for the whole data interval. Figure. 2 shows injection efficiencies of predicted by DNN (orange) and real (blue), for 2020/11/30-2020/12/07 evaluation data. Vertical and horizontal axis indicate injection efficiency and date, respectively.

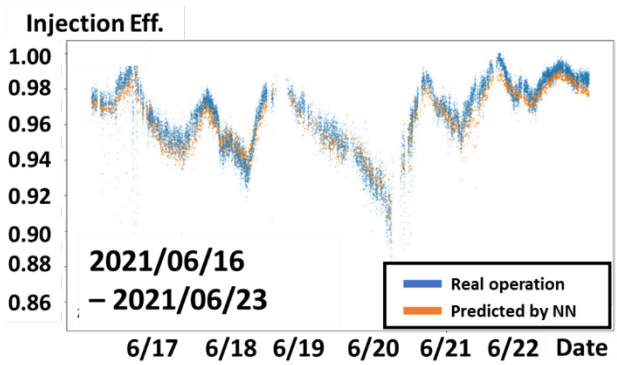


Figure 2: Injection efficiencies of predicted by DNN (orange) and real (blue), for 2020/11/30-2020/12/07 evaluation data. Vertical and horizontal axis indicate injection efficiency and date, respectively.

These results show that Regression-DNN can predict injection efficiency. In other words, Regression-DNN can describe the relationship accelerator parameters and injection efficiency.

Training DNN with past data Figure 1 shows that 2D latent value for 2021 Jun. is close to 2021 May. This indicates that the 2021 May accelerator data are similar to the 2021 Jun. accelerator data. So, using DNN trained with 2021 May data, we may be able to predict the injection efficiency in 2021 Jun. To test our hypotheses, we create three datasets with different inclusion periods. Figure 3 shows an overview of the three datasets.

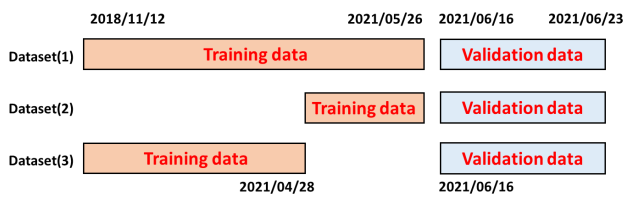


Figure 2: Overview of the datasets (1)-(3).

Each dataset contains 135,000 events for training and 15,000 events for validation. We trained and validated DNN on each dataset. As a result, the mean squared error of injection efficiency output by DNN trained on datasets (1)-(3) as validation data is 0.00038 (Dataset (1)), 0.00038 (Dataset (2)), 0.00300 (Dataset (3)). Figure 3 shows the comparison between the injection efficiency predicted by DNN (Dataset (1) orange, (2) red, (3) green) and the real

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

incident efficiency (blue). Vertical and horizontal axis indicate injection efficiency and date, respectively.

The results show that the DNN trained with 2021 May data; dataset (1,2), can predict 2021 Jun. injection eff. In other words, in order to predict the injection efficiency with DNN trained on past data, it is necessary that the training data of DNN contains similar data to the trend of the period we want to predict.

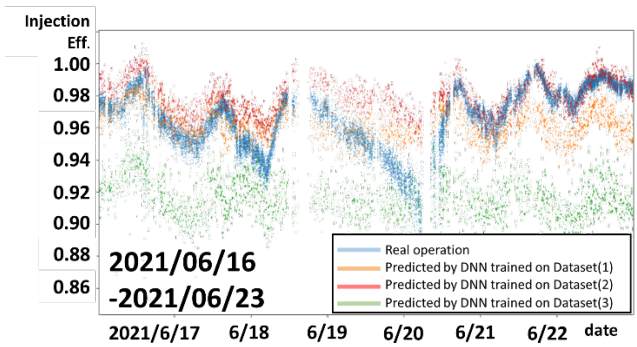


Figure 3: Predicted injection efficiencies for the 2021 June evaluation data, based on the DNN predictions with Dataset (1) to (3) (orange, red, and green), and the real operation injection eff.(blue). Vertical and horizontal axis indicate the injection eff. and date, respectively.

Accelerator Simulator Based on GAN

For the reinforcement learning, it is dangerous to do the parameter optimization based on the actual operation with the real accelerator. Therefore, realistic simulator is necessary for the pre-training of the actual reinforcement learning.

In this study, we develop a method using GAN (Generative Adversarial Network) [4]. Generative Adversarial Network (GAN) is a class of machine learning frameworks. Given a training set (real data), this technique learns to generate new data with the same statistics as the training set.

GAN consists of two networks (Generator and Discriminator) that compete with each other. The Generator is a network with noise data as input and new data with the same statistics as the training set as output. The Discriminator is a network with training set data(real) and output of Generator(fake) as input and discriminate between real and fake as output. By alternately training two competing networks, we can increase the similarity between the Generator output (fake data) and the training set (real data).

Using this property, by training GAN on the accelerator parameters data, it is expected to be possible to reproduce the data with keeping the correlation between the parameters. So, we verify whether the GAN trained by real data of the accelerator can generate data reproducing the correlation between parameters.

For GAN training, we create the following two datasets with different parameter configurations.

- Dataset(a) 2params
  Injection eff. + specific steering magnet param
- Dataset(b) 1233params
  Injection eff. + operating params+ env params

Please note that, Injection efficiencies in the datasets are uniformly extracted. And, each dataset contains 540,000 events for training and 60,000 events for validation.

Training by Dataset(a) We trained the GAN on the dataset (a), which consists of two parameters, the injection efficiency, extracted with a uniform distribution and specific steering magnet parameter (PY_32_4). Figure 4, shows a histogram of the magnet parameter (PY_32_4) reproduced by this trained GAN, and the real data used for training

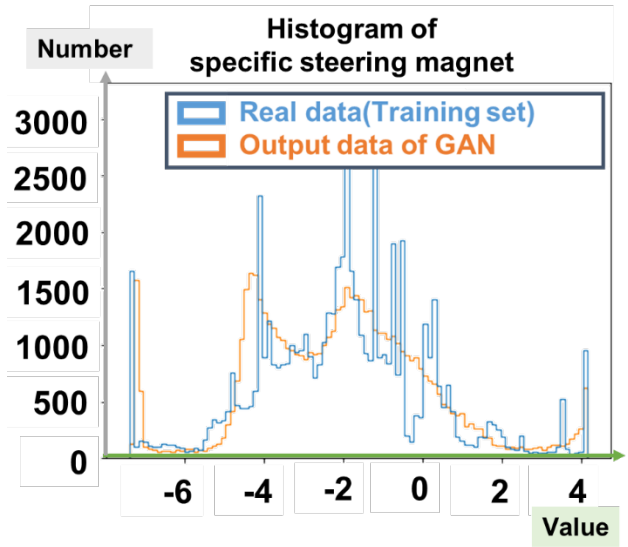


Figure 4: Histogram of the specific magnet (PY_32_4) param. reproduced by GAN trained Dataset(b) (orange) and the real data (blue).

In this case, from the results in Figure 4, we can see that GAN reproduces the characteristics (peaks, distribution) of real data by training. However, it is necessary to reproduce the correlation between the parameters as well as the characteristics of the individual parameters. Therefore, we examined the correlation between the injection efficiency and the steering magnet parameters in the dataset by making two-dimensional histograms. The created two-dimensional histogram is shown in Figure. 5.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV010

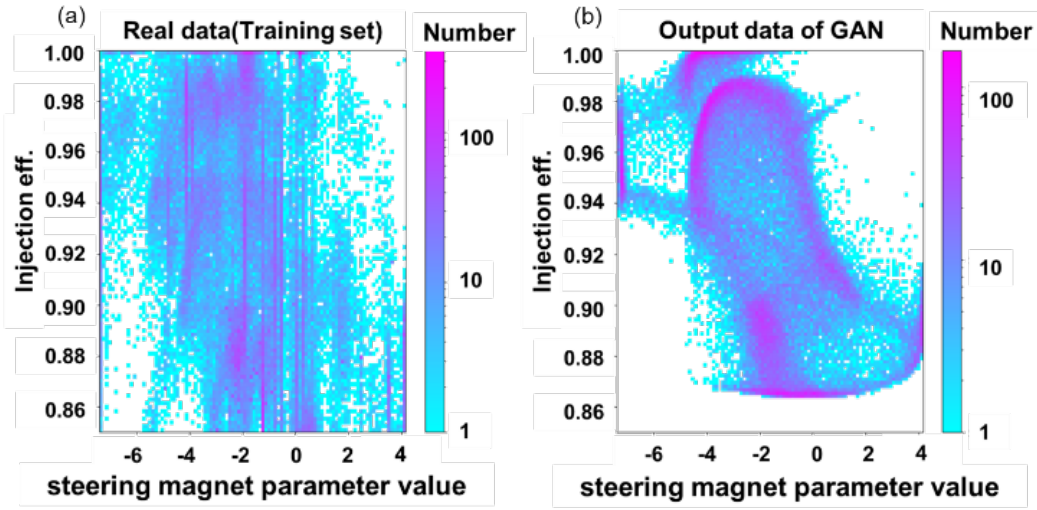


Figure 5: Two-dimensional histogram of real data(a) and reproduced data(b). Vertical and horizontal axis indicate Injection eff. and steering magnet parameter value.

From the results in Figure 5, we can see that GAN reproduces the two-parameter correlation trend. However, the GAN output data has peaks that did not match the real data and does not cover the full range of the real data.

Training by Dataset(b) We trained the GAN on the dataset (b), which consists of 1233 parameters, the injection efficiency extracted with a uniform distribution, steering magnet parameters and environmental parameters. Figure 6, shows a histogram of the magnet parameter (PY_32_4) reproduced by this trained GAN, and the real data used for training.

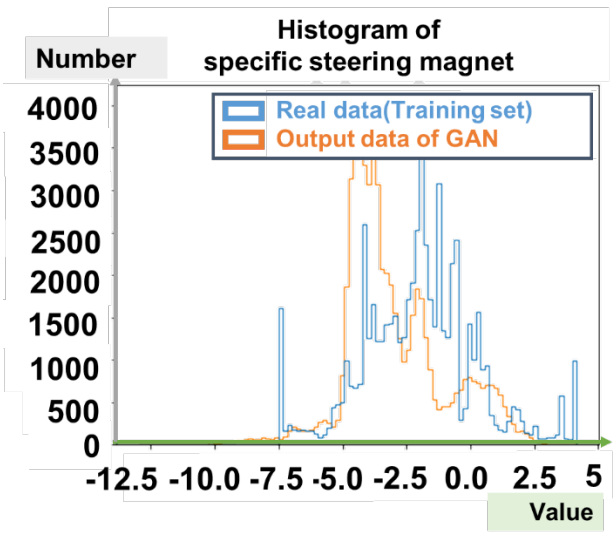


Figure 6: Histogram of the specific magnet (PY_32_4) param. reproduced by GAN trained Dataset(b) (orange) and the real data (blue).

From the results in Figure 6, we can see that GAN reproduces the characteristics (peaks, distribution) of real data by training. But we need to verify that the trained GAN is able to reproduce the correlations of the 1233 parameters, not just the individual parameters characteristics. Here, it is difficult to compare the real data with the reproduced data for the 1233 parameters. Therefore, we use the visualisation method with dimensionality reduction by VAE described above. Figure 7 shows two-dimensional histogram of the visualisation of the real data and reproduced data by GAN, using VAE trained by 2018 Nov to 2021 Jun data.

From the results in Figure 7, we can see that GAN output data dose not reproduce the whole period of the real data. Such a result, where the GAN can only reproduce part of the data used for training, is a possible event in the training of GAN, called "mode collapse".

SUMMARY

We have developed a machine-learning-based operation tuning scheme for the KEK e-/e+ injector linac (Linac), to improve the injection efficiency.

In this paper, we show the results on our R&Ds of, 1. visualization of the accelerator parameters (~1000) trend/correlation distribution based on the dimensionality reduction using VAE, 2. predicting injection efficiency using Regression-DNN, and 3. reproduction of accelerator parameters using GAN.

Using the electron beam data for Super KEKB injection accumulated in 2018 Nov. - 2021 Jun, we show that 1. It is possible to visualize the 1232 parameters by dimensionality reduction to two dimensions using VAE; 2. It is possible to predict the injection efficiency of accelerators using Regression-DNN. The prediction with the past data which has the similar parameter features, is effective to improve the prediction accuracy; 3. Accelerator simulator based on GAN reproduces the overall parameters correlation trend. On the other hand, GAN's output data dose not reproduce the whole period of the real data due to the "mode collapse", and further improvement is necessary to use for the pretraining of the reinforcement learning for the accelerator tuning system.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV010

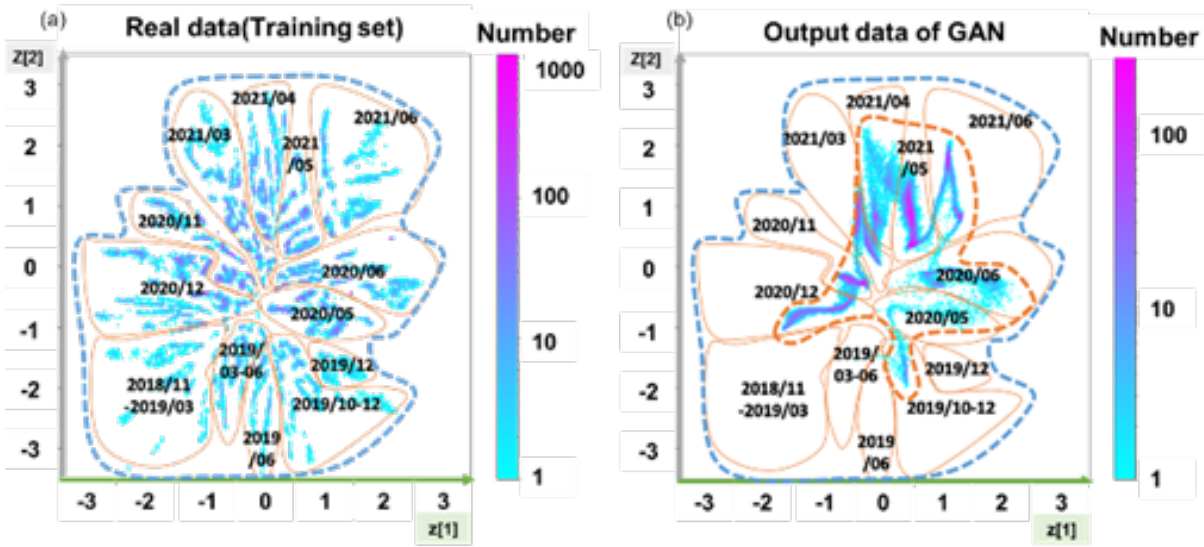


Figure 7: Two-dimensional histogram of real data(a) and reproduced data(b). Vertical and horizontal axis indicate VAE 2D latent value z [2] and z [1].

ACKNOWLEDGEMENTS

The R&D works are supported by the Research Project at the Research Center for Nuclear Physics (RCNP), Osaka University, and the Interdisciplinary Project of the Datability Frontier Organization, Osaka University. Part of this work is also supported by a collaborative research grant from the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT) under the "Diversity Research Environment Initiative (Traction Type)", and use of the Supercomputer System OCTOPUS at the Cybermedia Center, Osaka University.

REFERENCES

[1] Electron-positron injector (LINAC), https://www.kek.jp/en/Research/ACCL/LINAC/

[2] D. P. Kingma, and M. Welling, “Auto-Encoding Variational Bayes”, arXiv:1312.6114, 2014.

[3] TensorFlow, https://www.tensorflow.org/

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Networks”, arXiv:1406.2661, 2014.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV011

RESEARCH ON CORRECTION OF BEAM BETA FUNCTION OF HLS-II STORAGE RING BASED ON DEEP LEARNING

Yong-Bo Yu*, Ke Xuan, Gong-Fa Liu, Wei Xu,Chuan Li, Wei-Min Li
National Synchrotron Radiation Laboratory, University of Science and Technology of China, Hefei Anhui, China

Abstract

In recent years, artificial intelligence (AI) has experienced a renaissance in many fields. AI-based concepts are nature-inspired and can also be used in the area of accelerator controls. At HLS-II, there are not many studies on these procedures. We focused on HLS-II beam stability in order to get better performance. We have created a deep learning-based approach for correcting beta function. Simulation studies reveal that the method presented in this work performs well in terms of correction outcomes and efficiency, resulting in a new way to adjust the accelerator beam function.

INTRODUCTION

Since the 1980s, artificial intelligence (AI) approaches in accelerator control have been studied [1]. In the light of recent theoretical and practical advances in machine learning and the use of deep neural network-based modeling and controlling techniques, new approaches for the control and monitoring of particle accelerators are emerging. Furthermore, the availability of powerful deep learning programmings frameworks like TensorFlow [2], PyTorch [3], Keras [4], and Matlab allow rapid and optimized implementations of complex algorithms and network architectures. Therefore, we propose a method based on a deep neural network to correct the beta function.

FEEDBACK THEORY

Corrective Theory for the Beta Function

The beta function is the lateral dynamic function of the particle, and it is one of the most important optical parameters of a beam. The focus intensity K of the quadrupole and the change $\Delta Q_{x,y}$ of the storage ring tune at this time are recorded so as to calculate the beta function of the quadrupole. When changing ΔK, the theoretical formula for maintaining the measured value of the ring beta function is

$$\beta_{x,y} = \pm \frac{2}{\Delta K l}\Big(\cot\left(2\pi Q_{x,y}\right)\left[1-\cos\left(2\pi\Delta Q_{x,y}\right)\right] + \sin\left(2\pi Q_{x,y}\right)\Big) \quad (1)$$

The theoretical formula for the measured value of the function can be simplified as follows when the tune is far from the integer or half-integer resonance line, and the change value is small.

$$\beta_{x,y} \approx \pm 4\pi \frac{\Delta Q_{x,y}}{\Delta K l} \quad (2)$$

* yybnsrl@mail.ustc.edu.cn

According to the formula,the beta function at the location a quadrupole magnet is calculated using the variation of the quadrupole strength and the measured tune shift.

Using a Deep Learning Model to Conduct Beta Function Correction

The beta function of the storage ring receives feedback correction based on the generated storage ring beam function model. The feedback correction diagram of the HLS beam beta function is shown in Fig. 1. The steps involved in beta function feedback correction are followed: The focus intensity change of the storage ring quadrupole is obtained by feeding the beta function error value into the storage ring beam model. The focus intensity change is sent back to the storage ring to obtain the amended beta function value. This goes back and forth until the beta function is wholly corrected.

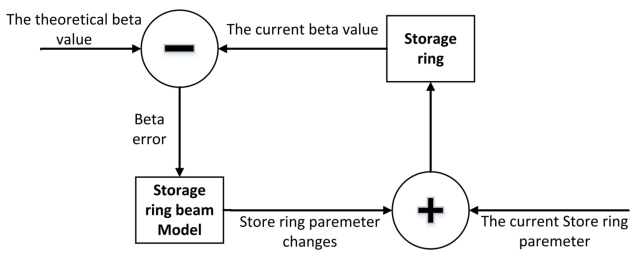


Figure 1: Schematic of the beta function correction system using a Neural network method.

MACHINE LEARNING DESIGN STEPS

Five significant steps are involved in developing an ML-based beat function application (see Fig. 2). Data acquisition and cleaning are the first steps. The topology of the neuron network is then defined and optimized. Finally, the beat function correction application must be tested after multiple training sessions and continuous performance tests. In the following sections, we will go over the most critical development steps.

Data Generation

In order to perform supervised neural network learning, A large number of data pairs must be provided. As a result, we took the Lattice as an object and created the HLS-II virtual storage ring model. The final data is finished with MATLAB's AT toolbox and a python program created with pyAT. After that, 10,000 data pairs were created. Each data pair has 96 data points, with Δbetax (32), Δbetay (32), and Δk (32).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

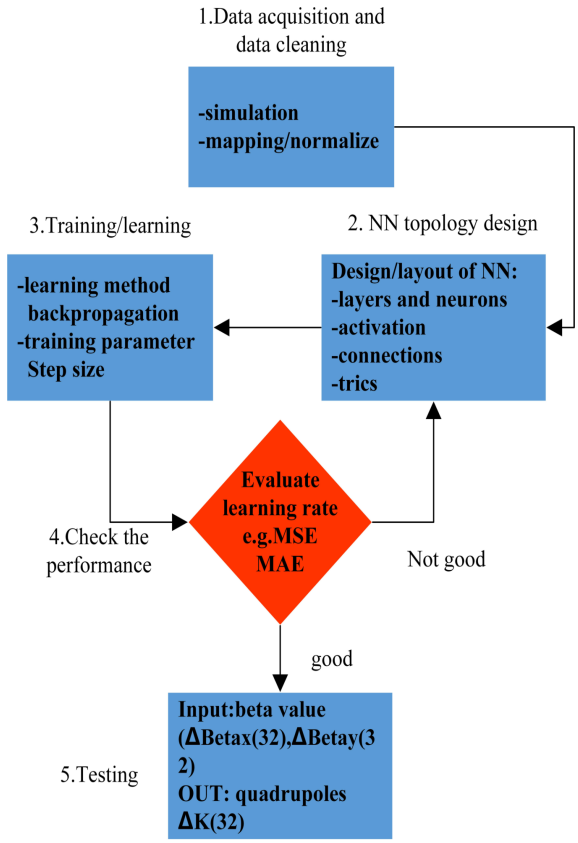


Figure 2: Development stages for an ML-based beta function correction.

Definition of Neural Network Structure

The topology diagram of the neural network after screening is illustrated in Fig. 3, which includes four hidden layers, batchnorm layers, and dropout layers. It has six layers of neurons, with 64 input neurons (related to the number of beta functions) and 32 output layer neurons (for the number of quadrupole values). The neurons in the buried layer are chosen based on previous experience [5]. Deep neural networks can approximate continuous functions of any complexity with arbitrary precision, according to research. For simulation, we use a deep feedforward neural network. The TensorFlow2.0 framework is used to create the deep neural network model.

Some Processing

Normalization is applied to both the input and output, and they are on a single uniform standard. Full connection is used in the neural network connection, and Dropout is used in this application. Dropout is a type of learning strategy that's used to improve neural networks. Dropout randomly disables a small number of neurons throughout each training phase, drastically reducing the complexity of the connection. It can also help to reduce overfitting caused by data while also improving the generalization of neural networks.

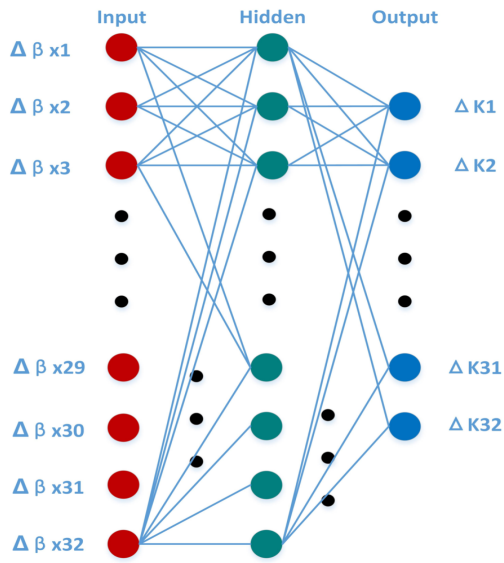


Figure 3: Feed Forward Neural Network (FFNN) topology used for ML-based beta function correction.

Model Train

The data comes from the previous virtual storage ring. A total of 1,000 groups are generated. These training data sets are divided into two parts in proportion.

1. data just for pure training, adjusting weights and bias values to minimize the MSE (80%)
2. Testing data is used to measure how accurately the network was trained (20%)

Adjusting the hyperparameters involves the first change, which is changing the learning rate. Let us use Adam, a gradient-based optimization approach rather than stochastic gradient descent. Because of his excellent computational efficiency and minimal memory footprint, Adam enjoys the advantages. It is pretty simple to use the hyperparameters, and just a few parameter tweaks result in the desired outcome. The iteration count is also critical. This is the number of times the entire training set is input to the neural network. Iterations can be considered adequate when the difference between test and training is minimal. It is over-fitting if the loss value initially decreases and then increases. The training times must be lowered. After testing, it was found that the number of epochs is 100 iterations, resulting in superior outcomes. The losses on the training set are known as "train loss" and on the test set as "value loss."MSE is a quantitative representation of model performance. So after around 80 iterations, the training set and test set stabilized. The MSE is less than 10-e5. Currently, it is confirmed that the neural network fitting algorithm's accuracy is acceptable, and the measured results are also rather dependable.

SIMULATION ANALYSIS

Therefore, we have corrected the beta function using a fairly complete neural network model so far. Through simulation, we generate beta function that have errors at random.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV011

The results are depicted in Fig. 4 using this Storage ring beam model. We see that after many corrections, the beta function close in on their theoretical values quickly. The method has been proven to be feasible. The beta functions of the storage ring was measured before and after the lattice correction for comparison [6]. A before-and-after comparison shows that the average beatx beating was 22.82% and 3.3%, respectively. However, due to the tiny fluctuation in vertical, the correction has not changed significantly.

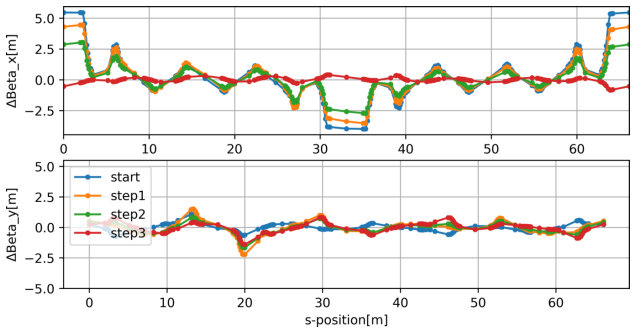


Figure 4: Beta parameters after iterations through the model.

CONCLUSION

The neural network approach is introduced to address the current beta function correction problem. Through the neural network model, the beta function is effectively corrected and has a good effect, providing fresh ideas and enhancing beam stability. Make a drawing. Other beam current properties on the storage ring can be studied using the neural network approach. It can be used to solve various optimization issues, including multi-parameter, multi-objective, so on and so forth.

ACKNOWLEDGMENT

The authors would like to thank all our colleagues at NSRL who provided us technical supports.

REFERENCES

[1] D. P. Weygland, "Artificial Intelligence and Accelerator Control", in *Proc. PAC'87*, Washington, DC, USA, pp. 564–566.

[2] TensorFlow, https://www.tensorflow.org

[3] PyTorch, http://pytorch.org/

[4] Keras, https://keras.io/

[5] D. Schirmer, "Orbit Correction With Machine Learning Techniques at The Synchrotron Light Source Delta", in *Proc. ICALEPCS'19*, New York, NY, USA, doi:10.18429/JACoW-ICALEPCS2019-WEPHA138

[6] K. Xuan, G. Liu, C. Li, J. Wang, L. Wang, W. Li, and J. Y. Li, "High Level Applications For HLS-II", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 974–976. doi:10.18429/JACoW-ICALEPCS2015-WEPGF117

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV012

BEAM FAST RECOVERY STUDY AND APPLICATION FOR CAFe

Jiaosai Li†, Youxin Chen, Jing Wang, Feng Yang, Hai Zheng, Institute of Modern Physics, Chinese Academy of Sciences (IMP/CAS), Lanzhou, China

Abstract

Based on the MASAR (MAchine Snapshot, Archiving, and Retrieve) system [1], a beam fast recovery system was designed and tested in CAFe (Chinese ADS Front-end Demo Superconducting Linac) at IMP/CAS for high current CW (Continuous Wave) beam. The proton beam was accelerated to about 20 MeV with 23 SC (Superconducting) cavities, and the maximum current reaches about 10 mA. The fast-recovery system plays a major role in the 100-hours-100-kW long-term test, during which the average time of the beam recovery is 7 seconds, achieving the availability higher than 90%. The system verifies the possibility for high current beam fast recovery in CiADS (China initiative Accelerator Driven sub-critical System).

INTRODUCTION

The ADS superconducting proton linac prototype (CAFe) led by the Institute of Modern Physics, Chinese Academy of Sciences has continuously improved technology and achieved remarkable innovation in commissioning and operation in recent years: 15~16 MeV@2mA CW proton beam stable operation up to 100 hours in the early of 2019 [2], 100 kW proton beam stable operation at 10 mA in CW mode at the beginning of 2021 [3]. The control system provides an important safeguard for CAFe linac during the stable operation. This paper introduces MASAR and its application in CAFe linac, mainly concerning beam fast retuning.

MASAR

MASAR is an epics V4 service, which is developed based on C++ and python. It was originally proposed and developed by Brookhaven National Laboratory in the United States, and then applied to the second generation synchrotron radiation light source (NSLS-II). The function is just like its name, including the archiving, comparison and restoration of machine snapshots, data archiving and data retrieval [1].

The MASAR consists of client and server: the client has a client Python API library, which can be used by both Python scripting and the GUI. A default PyQt4 GUI is developed for data viewing, snapshot taking, comparing a snapshot with a live machine, and restoring the machine to a particular status using a snapshot. The server has 4 layers: Service communication control; Service: parses a command from the client, implements the desired action, and returns the result to the client; Channel Access Client; DSL (data source layer) and Data layer [3][4]. Figure 1 shows the MASAR Architecture.

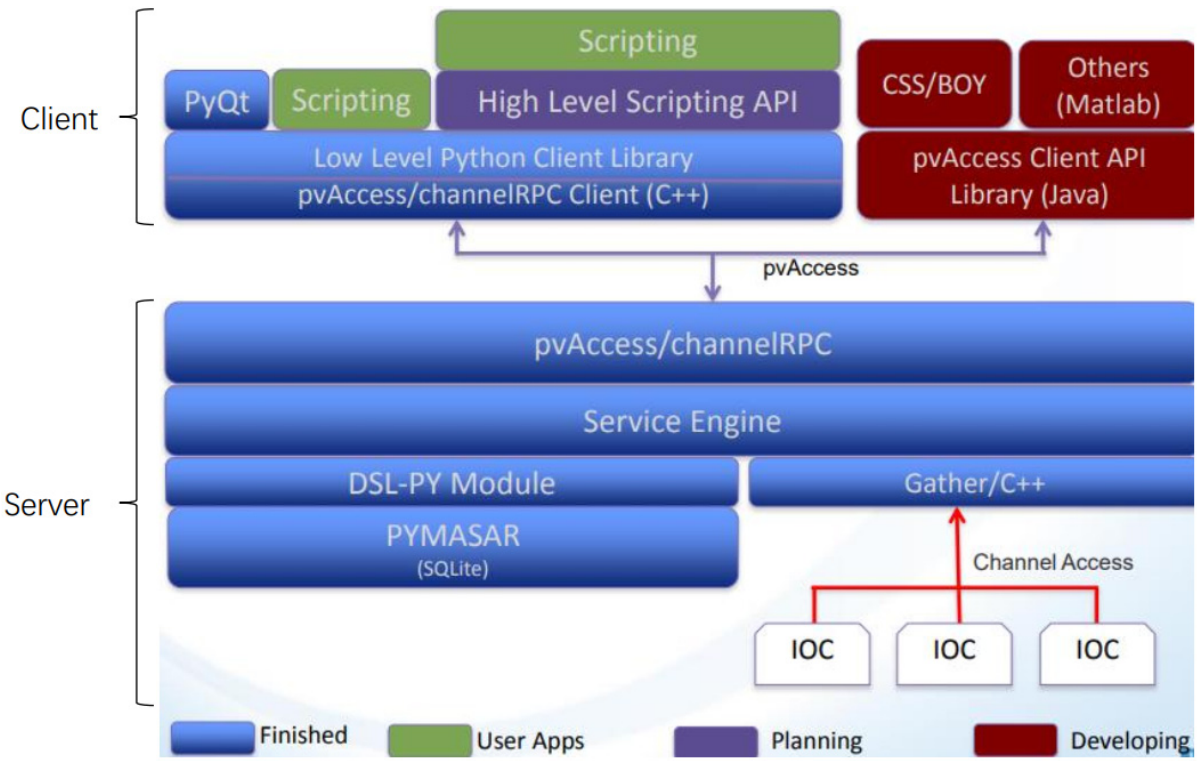


Figure 1: MASAR Architecture.

† ljs@impcas.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV012

BEAM FAST RECOVERY EXPERIMENT BASED ON MASAR

The purpose of beam fast recovery is, when faults occur to the machine, the beam is recovered in a short time. The machine faults should be judged before the recovery action by MPS: if the faults are serious, like severe LLRF (Low-Level Radio Frequency) failure, then the beam should be stopped before the RFQ (Radio Frequency Quadrupole); otherwise, the beam fast recovery system takes action. The beam fast recovery system helps improve the availability of the machine and ensures the stable operation of the beam for a long time.

Considering various recoverable faults, a series of orderly automatic recovery measures are taken. When BPMs (Beam Position Monitor) detect abnormal beam status, the beam will be stopped quickly; when all system returns to normal, the beam will be quickly restored.

The beam fast recovery system includes the operation, monitoring and judgment of MPS (Machine Protection System), BPS (Bypass Control System), timing and LLRF systems. The optimization of existing hardware improves the reliability and recovery speed, and main manual operation including pulling out the FC (Faraday Cup) before RFQ is replaced by the control system, without losing machine safety at the same time.

During beam recovery, it is necessary to judge the beam status (it is probe beam or continuous wave beam) and BPM status (the phase and position of the beam at all BPMs). Therefore, before the recovery, the machine snapshot should be taken when the machine operates normally and stably (beam operation leader confirms that the beam reaches normal status), and the 81 BPMs snapshots (27 BPMs in total, each consists of Phase, X position and Y position) are saved, which is treated as the standard for BPM comparison and judgment. To ensure that the phase and position information of the beam at BPMs are saved at the same time, MASAR needs to be used for one-click operation.

When BPM judgment starts, the PV should be judged whether it is disconnecting. If so, set disconnect_Flag to true, otherwise false. The trigger number (trig_No.) is initialized to 0.

If trig_No.<3, Count (the number of PVs' value satisfied the conditions) is initialized to 0. If BPM's PV is bypassed, Count plus 1; otherwise, the PVs' value is judged whether within the standard threshold. If within, Count plus 1 and Flag is set to true; if not within, F_Count (F_Count indicates the number of PVs' value not satisfied conditions) plus 1 and Flag is set to False. Then, judge F_Count > 6, if not, judge next PV is bypassed. If so, break off PV judgment.

Then judge whether count>=76: if so, return 1, indicating that the judgment passed; otherwise, Trig_No. plus 1.

If Trig_No.>=3, check disconnect_ Flag is true or not. If true, return -1, the reason for not passing the judgment is that PV disconnect. If not, return 0, the reason for not passing the judgment is the number of PV's value satisfied conditions is less than required. Figure 2 shows the BPM judgment process.

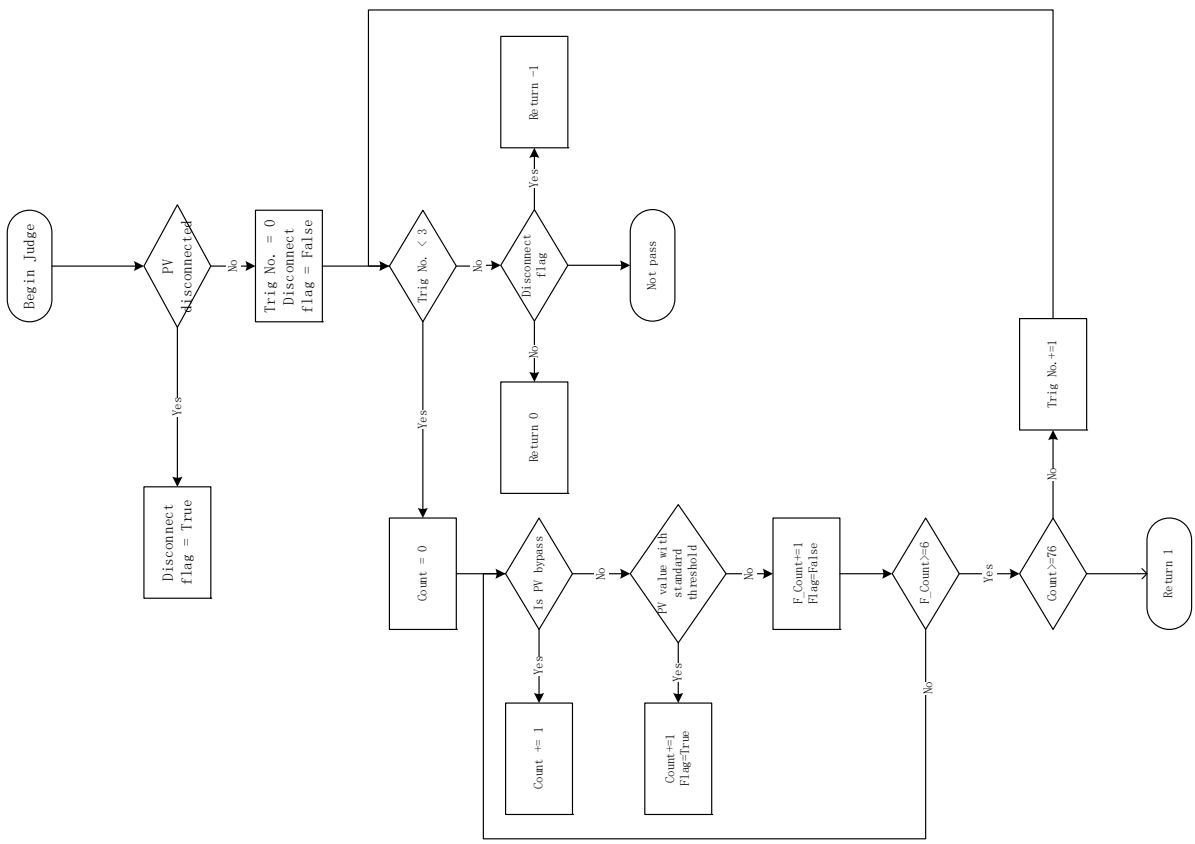


Figure 2: BPM judgment process.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

We use MASAR to save the relevant information of BPM beam: Firstly, save the PV list of 81 beam phase and position information of BPM in masarServerce/example/pvs/beam_auto_Re.txt text file; Secondly, configure the PV group of MASAR and establish a connection with the PV related to publishing BPM, so that the PV value can be obtained through CA protocol when it is necessary. Edit the masarService/Python/pymasarsqlite/db/settings.py file and add the following content:

```
# pv group name: [pv list file, description]
pvgroups= {
    'BEAM_AUTO_RE': ['pvs/beam_auto_re.txt', 'beam auto recovery use snap give bpm']
}
# config name: [config desc, system]
configs= {
    'BEAM_AUTO_RE': ['beam auto recovery use snap give bpm', 'BM']
}
# config name: [pvgroup,]
pvg2config= {
    'BEAM_AUTO_RE': ['BEAM_AUTO_RE']
}
```

Finally, create the SQLite3 database: Python masarService/python/pymasarsqlite/db/configmasardb.py. So far, the BPM snapshot of MASAR has been configured. Restart the MASAR server:

```
$ cd masarService/cpp
$ ./bin/linux-x86_64/masarServiceRun
```

The interface is shown in Figure 3.

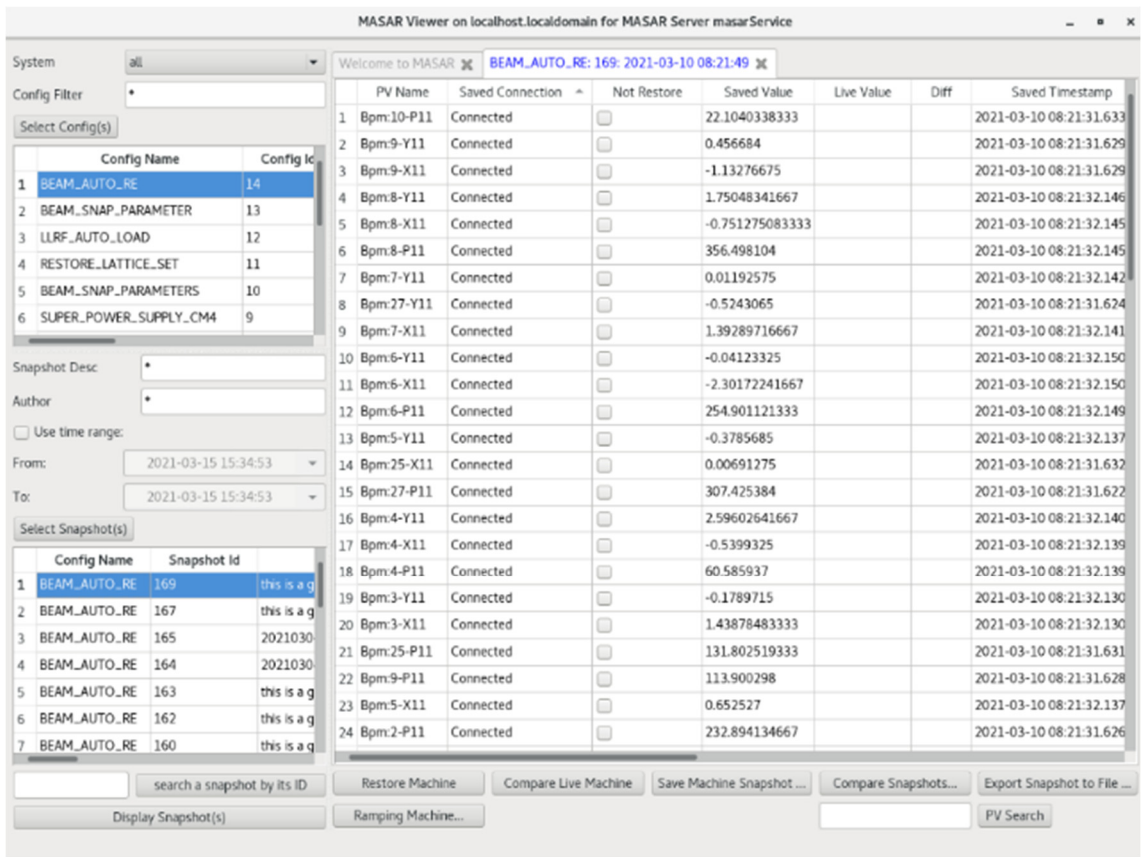


Figure 3: MASAR view

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV012

According to the operation status of the machine, the beam operation leader judges when the beam reaches a normal state, and the MASAR snapshot is saved with the beam phase and position information at all BPM. Call the saveSnapshot() function of MASAR, and use it as the judgment standard of BPM beam phase and position information during beam fast recovery. Before the subsequent machine operation, retrieve the beam phase and position information from the previously saved BPM data, then call the retrieveSnapshot() function, and assign the saved PV values of the BPM phase and position to the standard PV values of the corresponding BPM phase and position one by one, for the comparison in the beam fast recovery experiment.

CONCLUSION AND PROSPECT

In the experiment, as a tool of EPICS, MASAR realized fast saving snapshot, retrieving snapshot data and assisting snapshot data value recovery in 1 s, which leaves enough time for the adjustment of the hardware system, shortened the beam recovery time and improved the availability of the machine. The results of the 100-hours-100-kW beam stability test are shown in Table 1, and the availability of the machine is up to 93.5% with beam fast recovery. The beam recovery time distribution is shown in Figure 4, and recovery issues within 6 s take higher than 80%. The frequency statistics for automatic recovery are shown in Figure 5, of which 90% have the condition for automatic recovery and 78% are successful. The system verifies the possibility for high current beam fast recovery in CiADS.

Table 1: Test Results

Parameters	Value	Unit
Beam energy	17.272±0.034	MeV
Average current intensity	7.2966±0.0244	mA
Beam power	126.0	kW
Total test time	108	h
RF superconducting system availability	98	%
Machine availability	93.5	%

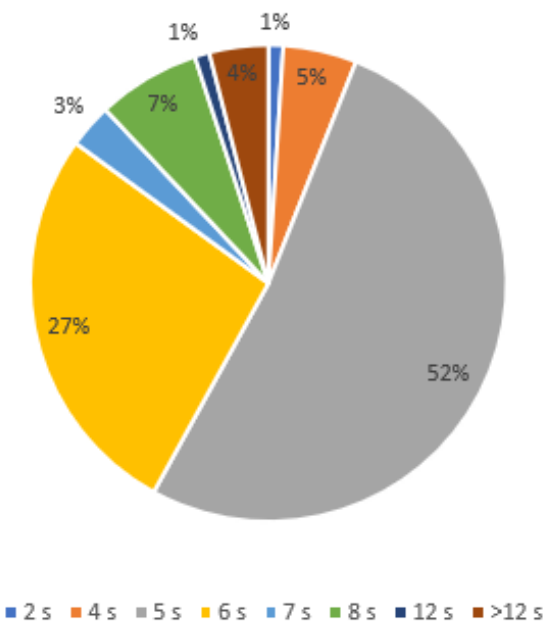


Figure 4: The time required for beam recovery.

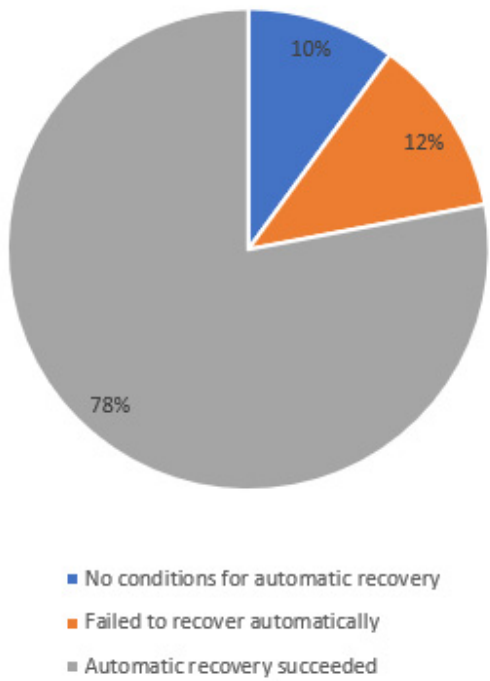


Figure 5: The automatic recovery times.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

There are many advantages in beam fast recovery system: compared with manual recovery, the time is very short; automatically judge the beam status, which is more reliable; automatic execution, greatly reducing labor intensity and improving reaction speed; most trip situations could be judged and handled.

There are still some issues that need to be optimized in the beam fast recovery system. PV disconnection occasionally occurs during the experiment, and with the number of disconnection times increases, the time required for reconnection becomes longer, which is related to the reconnection mechanism of PV. In addition, the beam fast recovery system is not stable enough, and the stability of the program needs to be improved.

ACKNOWLEDGEMENTS

The authors would like to thank the linac center in IMP/CAS, especially the control group for their help.

REFERENCE

[1] Guobao Shen, Marty Kraimer, "MASAR USER MANUAL", BNL, 2012.
http://epics-pvdata.hg.sourceforge.net/hgweb/epics-pvdata/masarService/raw-file/tip/documentation/userManual.html

[2] Linac Center of IMP/CAS, "ADS Proton Superconducting Linac realized 15~16 MeV/2 mA 100 h CW operation", 2019-01-28.
http://www.impcas.ac.cn/kyjz2017/201901/t20190128_5235732.html

[3] Linac Center of IMP/CAS, "IMP High Current Superconducting Proton Linac Passed The Expert Group Test", 2021-03-11.
http://www.impcas.ac.cn/kyjz2017/202103/t20210311_5973343.html

[4] Guobao Shen, "MASAR Service", BNL, Oct 05, 2013.
https://epics.anl.gov/meetings/2013-10/3%20-%20Other%20Services/4%20-%20MASAR%20Service.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV013

DESIGN OF MAGNET MEASUREMENT SYSTEM BASED ON MULTI-HALL SENSOR

B. J. Wang*, Y. H. Guo, N. Xie, R. Wang
Institute of Modern Physics, Lanzhou, China, 730000

Abstract

High-precision magnetic field measurement and control technique significantly guarantees the accurate realization of the magnetic confinement of accelerators. Using real-time magnetic field intensity as the feedback to adjust the magnetic field current input can be a promising strategy. However, the measurement accuracy of the Hall-sensor is hard to meet feedback requirements because of multiple affection from external factors. Meanwhile, the NMR(Nuclear Magnetic Resonance sensor), which can provide high-precision magnetic field measurement, can hardly meet the requirements against the real-time control due to its strict requirements on the uniformity of the measured magnetic field, as well as its low data acquisition speed. Therefore, a magnetic field measurement system based on multi-Hall sensors is designed to solve this problem. Four Hall-sensors are used to measure the target magnetic field in this system. An Adaptive fusion algorithm is used to fused collected values to obtain the best estimate of the magnetic field intensity. This system effectively improves the accuracy of magnetic field measurement and ensures the instantaneity of the measurement.

INTRODUCTION

Magnetic confinement plays a decisive role when stabilizing the accelerator beam. At present, the control of magnetic confinement is mostly based on the feedback of the power supply current to implement of the current input of magnet [1], which indirectly ensures the stability of the magnetic field. However, this method is difficult to ensures the effect of the control of the magnetic field, due to magnet's own factors such as magnet eddy current and iron core aging. Therefore, directly using magnetic field strength, as a feedback signal becomes a more promising solution. Nevertheless, it is limited by the sampling rate and accuracy of magnetic field measurement, which makes the progress of this solution relatively slow.

Present main methods of magnetic field measurement and their accuracy are shown in Fig. 1 [2]. The feedback signal of magnetic field control requires high precision and wide range of measurement. So, NMR, hall, and fluxmeter meets the requirement. However, the NMR sampling frequency is than expected, and it needs a certain period of time to lock the value of non-uniform magnetic field. For instance, the sampling frequency of Metrolab PT2026 is only 33 Hz, and the search time is approx [3]. Thus, NMR is difficult to meet the magnetic field control requirements. Fluxmeter is commonly used for dynamic measurement. But its size is large, which makes it difficult to be broadly deployed on site. Hall sensor advantages of high sampling rate, excellent dynamic characteristics, small volume and easy deployment. So, it is the most suitable magnetic field measurement scheme for feedback control.

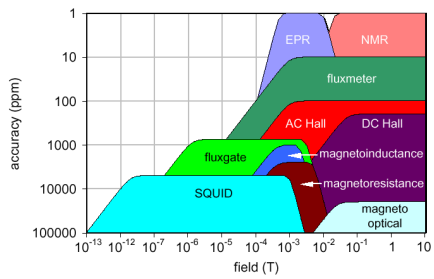


Figure 1: Magnetic field measurement range and accuracy.

SYSTEM DESIGN

Signal Preprocessing

The main noise of the measurement system includes the offset voltage and thermal error of Hall element [4], as well as the gain error, offset voltage and accidental high-frequency noise of the measurement module [5]. To eliminate these noises, the single-channel measurement signal is preprocessed. The high-frequency noise and offset voltage are mainly filtered by pre-filter and differential algorithm within the measurement module. The pre-filter distinguishes signals according to the frequency of the signal and blocks the high-frequency noise. On this basis, differential operation is performed on the input signal. The offset voltage of each ADC in the measurement module is approximately same. Equations (1) and (2) represents signal V_m.

$$V_{m^+} = V_{id} + V_{of} \tag{1}$$

$$V_{m^-} = -V_{id} + V_{of} \tag{2}$$

Where V_{of} is the offset voltage, V_{id} is the ideal signal. V_{m^+} and V_{m^-} represents a pair of measurements with opposite phases.

As shown in Eq. (3), an ideal input signal without offset voltage can be obtained by subtracting the input with opposite phase.

$$V_{m^+} - V_{m^-} = 2V_{id} \tag{3}$$

For Hall thermal error and measurement module gain error, the corresponding calibration parameters need to be obtained and compensated by software. Calibrating gain

* wangbaojia@impcas.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

error of measurement module with high precision calibration source. By comparing the calibration source with the measured voltage, the compensation coefficient α can be obtained. For the compensation of the Hall thermal error, it needs to be divided into two situations to calibrate the correction coefficient of Hall which are constant temperature with plus different magnetic field and constant field with plus different temperature. The correction value corresponding to different gradient magnetic fields at 25 °C is shown in Fig. 2.

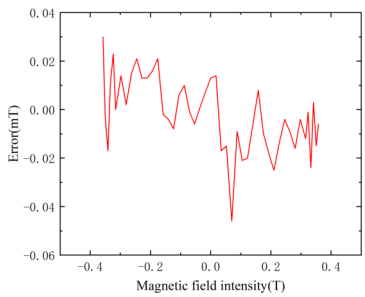


Figure 2: the correction value corresponding to different gradient magnetic fields at 25 °C.

Adaptive Weighted Fusion Algorithm

The offset voltage of Hall element is a kind of random noise which is it difficult to remove through common frequency-domain filter. Therefore, the offset voltage needs to be filtered by time-domain method. For a single sensor, the variance of the sensor cannot be changed, and the posterior error can be only reduced by increasing the measurement data. But this method increases the amount of computation and reduces the convergence rate. The method of multi-sensor adaptive fusion will not only improve the accuracy of measurement value, but also reduce the amount of measurement and accelerate the convergence rate.

For the adaptive weighted fusion algorithm, the fused value and adaptive weighting factor shall meet Eq. (4).

$$X = \sum_{n=1}^{4} W_n X_n, \sum_{n=1}^{4} W_n = 1 \tag{4}$$

Where X, X_n and W_n are respectively fused value, measured value of hall sensors and adaptive weighting factor.

The measured value of any sensor has the structure shown in Eq. (5).

$$X_n = ide_n + nis_n \tag{5}$$

Where the ide_n is measured value without noise. nis_n is offset noise, its variance is σ_n^2.

Then the weighting factor should satisfy Eq. (6). After weighting, the observation noise should be 0 or *constant*. So that the signal exclude noise can be obtained directly through linear correction. .

$$\sum_{n=1}^{4} W_n nis_n = 0 \; or \; constant \tag{6}$$

From the analysis above, the optimal weighting factor is determined by the variance of each sensor. However, the variance of sensors is often uncertain and will be affected by many factors. Therefore, directly calculating the sensor variance through the measured value and adaptively changing the weighting factor can reduce the noise to the greatest extent. When there are multiple sensors, the measured data can be used to obtain [6]. Take Hall sensors 1 and 2 as an example, where k is the sampling period, and the solution formula is represented as Eq. (7).

$$\sigma_1^2 = R_{1,1} - R_{1,2} \tag{7}$$

The $R_{1,1}$, $R_{1,2}$ solution algorithm is shown in Eqs. (8) and (9).

$$R_{1,1}(k) = \frac{k-1}{k} R_{1,1}(k-1) + \frac{1}{k} X_1(k) X_1(k) \tag{8}$$

$$R_{1,2}(k) = \frac{k-1}{k} R_{1,2}(k-1) + \frac{1}{k} X_1(k) X_2(k) \tag{9}$$

And through δ^2 can be solved the optimal weighting factor, where p is one of the four sensors, and its algorithm is shown in Eq. (10).

$$W_p = 1/(\delta_p^2 \sum_{n=1}^{4} \frac{1}{\delta_n^2}) \tag{10}$$

Thus, an adaptive fusion filter for four-way hall sensors is implemented. After the filtered signal is calibrated by NMR, the magnetic field intensity with few noise can be obtained.

Testing Platform Construction

The overall structure of the system is shown in Fig. 3. A standard dipole magnet is used as the detection object and NMR is used as the measurement datum. At the center of the pole of the bipolar magnet, four Hall sensors are deployed on both sides of the NMR probe. NI CompactRIO (CRIO) is used for data acquisition and processing of four-way Hall sensors. The host is responsible for collecting NMR and processed Hall data. And it also has the function of adjusting the power supply output to control the magnetic field.

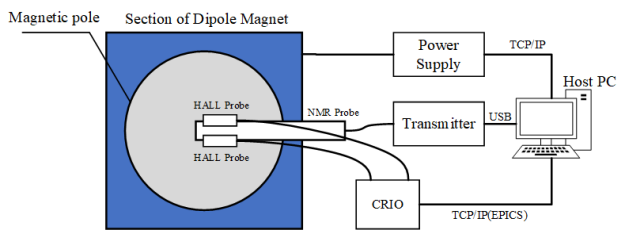


Figure 3: Overall structure of multi-hall magnetic field measurement experimental platform.

The host connects with CRIO and power supply through TCP/IP and carries out commands such as data reading and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV013

current output. The CRIO runs the EPICS program to publishes PVs. PVs includes four HALL readings and the magnetic field values after algorithm processing. Magnetic field measurements of NMR are directly transferred to the host through USB.

SIMULATION TESTING

Four independent groups of zero mean noise were used to simulate the observation error of hall sensor. The variance of white noise in each group were 0.001, 0.002, 0.0025, 0.0015, and the true value was 0.35 T. True value and white noise were added together to simulate four groups of sensor measurement data.

Figure 4 shows measured value of hall sensor with minimum variance and adaptive fused value. The measurement value of a single Hall sensor fluctuates around 0.35 T. The estimated value of adaptive fusion will have a short-term drift at the beginning due to insufficient data. As the number of data increases, the magnetic field data will become stable and converge to the true value.

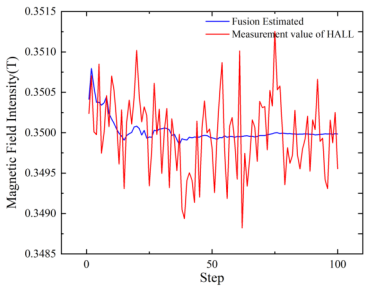


Figure 4: Comparison of minimum variance Hall magnetic field data with the estimated value after adaptive fusion.

The result of comparing errors between measured value of Hall sensor and fused value shows that the adaptive fusion algorithm can greatly reduce the measurement errors. As shown in Fig. 5, after using the adaptive filtering algorithm, the error quickly converges to the minimum with the step size. After 20 steps of data fusion, the error converges to 0. Accurate measurement of magnetic field is achieved.

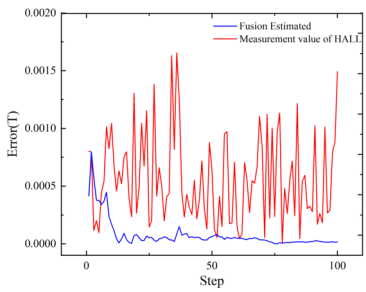


Figure 5: The estimation error of adaptive fusion and the error of Hall measurement with minimum variance.

In Fig. 6, the mean filter was a method that directly averaging the measurements of the four sensors. The effect

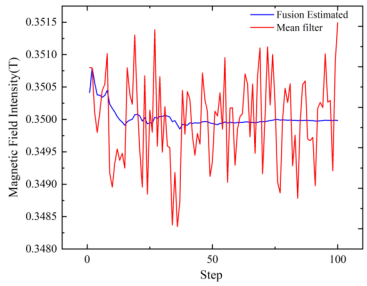


Figure 6: Comparison of adaptive fusion filter and mean filter.

of mean filter is obviously worse than that of the adaptive fusion algorithm because it simply sets the weighting factor of all Hall sensor measurements to 0.25.

CONCLUSION

In this paper, a measurement system is designed with multiple Hall sensor. A filtering scheme is designed for the noise of hall and measurement module. It includes pre-processing the HALL measurement value, which are weakening the high-frequency noise and offset voltage through filter and differential algorithm. The correction of thermal error of hall sensor and gain error of measurement module is also implemented. Eventually, the offset voltage of Hall sensor is eliminated by the adaptive fusion algorithm of multi-hall sensor. The final simulation results show that the measurement error of the magnet measurement system can be significantly reduced by using the adaptive fusion algorithm of multi-hall sensors.

REFERENCES

[1] Y. Kurimoto, Y. Morita, S. Nakamura and T. Shimogawa, "Precise Current Control in Accelerator Magnets with a Digital Feedback System", *IEEE Transactions on Nuclear Science*, vol. 61, no. 1, pp. 546–552, 2014.

[2] L. Bottura and K. N. Henrichsen, "Field Measurements", *Accelerators and Storage Rings CAS - CERN Accelerator School: Superconductivity and Cryogenics for Accelerators and Detectors*, pp. 118-148.

[3] PT2026 Precision Teslameter, `https://www.metrolab.com/zh/products/products-pt2026-nmr-precision-teslameter/`.

[4] J. E. Lenz, "A Review of Magnetic Sensors", *Proceedings of the IEEE 78*, pp. 973-989, 1990.

[5] F. C. Alegria and H. P. Silva, "Choosing Between Terminal and Independently Based Gain and Offset Error in the ADC Histogram Test", *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 1, pp. 9–16, 2011.

[6] S. L. Zhai, Y. S. Dai, "Study of Adaptive Weighted Fusion Estimated Algorithm of Multi-sensor Data", *ACTA METROLOGICA SINICA*, vol. 19, no. 1, pp. 69–74, 1998.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV015

DEVELOPMENT OF THE RF PHASE SCAN APPLICATION FOR THE BEAM ENERGY MEASUREMENT AT KOMAC*

SungYun Cho†, Jeong-Jeung Dang, Jae-Ha Kim, Young-Gi Song
Korea Multi-purpose Accelerator Complex, Korea Atomic Energy Research Institute, Gyeongju, Korea

Abstract

The Korea Multi-purpose Accelerator Complex (KOMAC) has been operating the 100MeV proton linear accelerator. The output beam energy from each drift tube linac (DTL) can be changed by the operation RF phase. The KOMAC linac is consisted of 11 tanks that need to proper setting of RF phase. The phase scan application was developed on the Java eclipse. On the other hand, the analysis program was developed on the Matlab. Since the data analysis processes as soon as finished the scan processing, the application integration need. This paper describes the implementation of the integrated application based on python and Experimental Physics and Industrial Control System (EPICS).

INTRODUCTION

The KOMAC control system has been operating based on the EPICS framework [1]. The distributed system that is an input output controller (IOC) controls the subsystem for each local system. The process variables (PVs) are operation parameters for control and monitoring from the IOC. Several programs can be access to EPICS IOC's parameters by channel access (CA). The RF phase scan application has been developed based on python and can be access to IOC using pyepics library [2]. The scan I/O variables were designed using that. The data processing step has been developed based on python numba library [3]. A scan step and data processing are described separately in order. The below figure is the schema regarding the KOMAC control system.

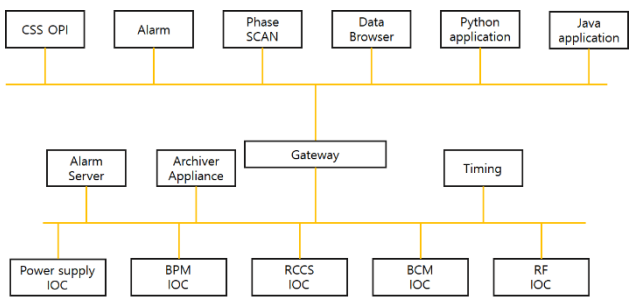


Figure 1: KOMAC distributed control system.

THE STRUCTURE OF THE PHASE SCAN APPLICATION

The basic structure of the application is based on Python Display Manager (PyDM) [4]. The PyDM is a building user interfaces tool based on PyQT5 with plugged in pyepics. The PyDM uses Signal & Slot for calling events. The I/O variables are connected with Slot functions. Some calculation function are developed based on numba. Figure 2 shows the working structure of the phase scan application.

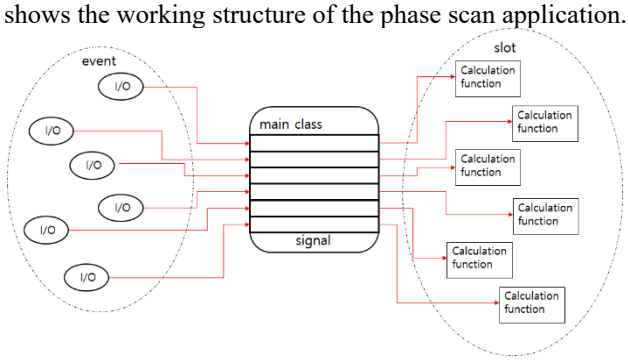


Figure 2: The mechanism of the application.

SCAN CODE DEVELOPMENT

The moving parameter is a phase set value that is different for each tank. After the parameters regarding scan settings are determined, the phase measure experiment will start. The measured phases are from the prior tank and current tank. So the moving parameter is just one and the monitoring parameters are two. Figure 3 shows the concept of process.

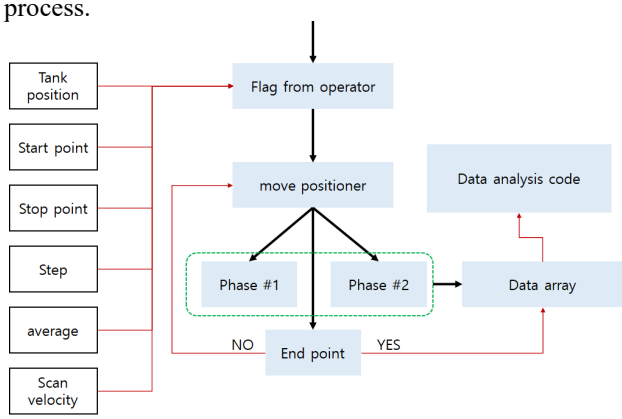


Figure 3: The schematic of scan algorithm.

The original program was developed by the Java language. Since the scan mechanism has to be transplanted to the new program, the EPICS sscan record that's similar to the original algorithm was selected [5]. The sscan record has a function to move positioners and record detector data at each of the positions. The positioner is the moving parameter for the phase setting. When the EXSC field that's a flag signal is executed, sscan record start the process. Many fields work organically during the process. The output signal is sent to positioner PV. If the changing value is finished, the sscan waits the delay time by PDLY field and the detector PV records the target value. The moving step

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV015

is determined by the NPTS field. The step can be changed by the custom function using calcout record, since the NPTS is the number of points in a range of between the P1SP and P1EP fields. The monitoring data is saved on the ai record. I/O signals control and monitor the PVs via pyepics library. Figure 4 shows the I/O signals with the IOC.

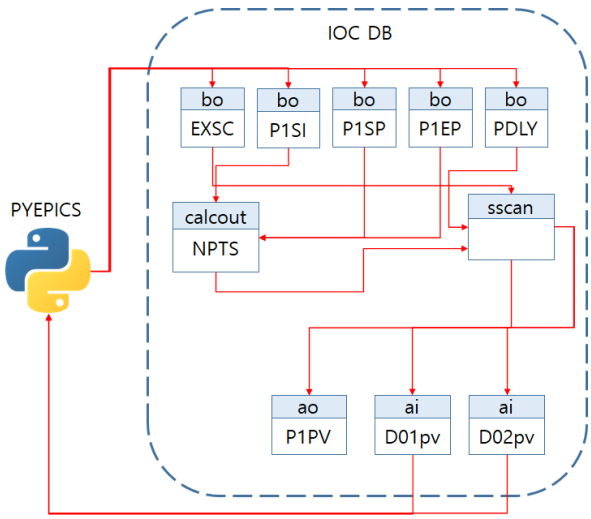


Figure 4: I/O signals from the python application.

The sscan record access to ao record via Channel Access because the set variable works other IOC. When the scan is finished, the monitoring data group is saved to the array and it is output to the python application.

THE DATA ANALYSIS

Figure 5 is overall schematic regarding the system mechanism.

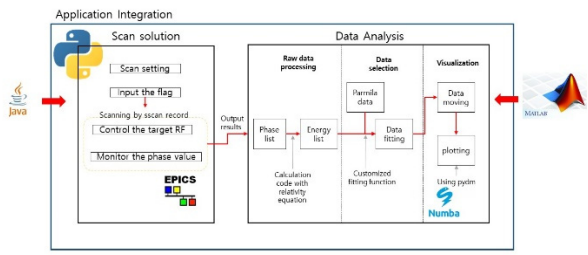


Figure 5: The schematic of the application integration.

The original analysis code was developed by the Matlab. The goal of development is high implementation and speed similar to the original. Because Matlab can provide a rapid calculation process more than python language, the calculation process needs to be rapid flatform. The numba library is selected because of it. The numba is a just-in-time compiler for python, it is suited to use with the numpy arrays and functions, and loops.

If the data inputs to the application, the data processing is started. Accumulated data is the phase values that are each target tank and prior tank. The first data processing is that phase values convert to energy data. The velocity of the beam can be calculated from the phase difference. The energy of the beam can be calculated according to einstein's equation. Figure 6 describes the phase scan method [6].

$$\mathrm{KE} = (\gamma - 1)\mathrm{mc}^2, \gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

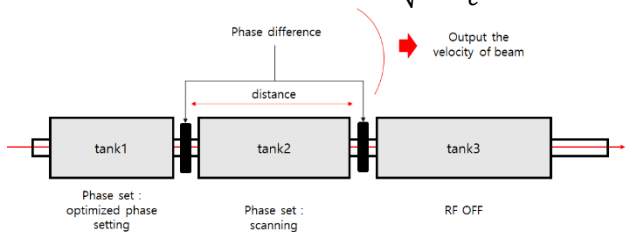


Figure 6: The phase scan signature matching.

The diagnostic device is a beam position monitor (BPM) that has offset value because it isn't synchronous with timing system. So each measurement point has to be calculated considering the offset value between -180 and 180 degree. It is implemented with 360 loops for each step based on numba environment.

After the energy list is accumulated, it can be called measured results. The measured results need to compare with the calculated results in order to determine the proper RF operating point. The calculated results can get from PARMILA calculation. Because the calculated results have to match the number of data for comparison, the data is selected the same RF operation phase with measured results. The optimized RF operation phase can be determined from the minimum chi-square value. Chi-squared test is the method that is determining whether the difference between the observed data with expected data.

$$\chi^2 = \sum \frac{(observed\ value - expected\ value)^2}{expected\ value}$$

The calculated results have several data group depending on RF amplitude. The result of chi-square is for each amplitude of parmila data. The results can be fitted to a quadratic equation using scipy library that provides the curve-fit function. Figure 7 shows the result of fitting.

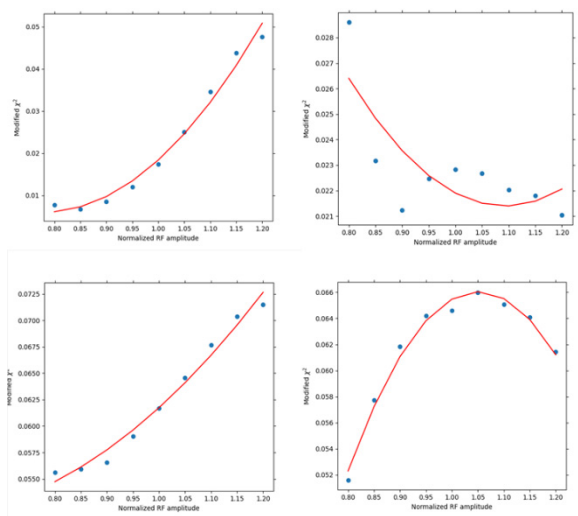


Figure 7: The results of the fitting.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

However, if the scipy library is used, the calculation speed is slow remarkably. Since the scipy library can't be used in the environment of numba, the custom fitting function needs to be built. The fitting function can be similar to the shape of a general equation. The quadratic gradient can determine whether it is concave or convex. The RF operation phase is selected when the result of fitting is a concave. The key of time reduction is just to do discrimination action in the loop except set up an equation. This method is approximately 200 times rapidly more than scipy function on the numba environment. But the overall result is a little broadly than the original result. Figure 8 shows the result of comparing.

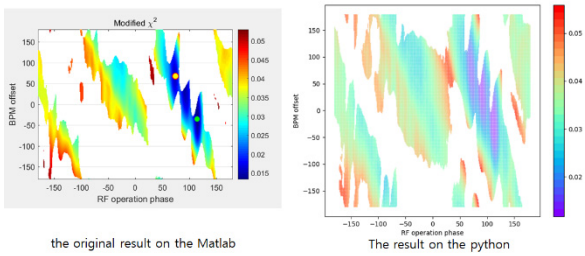


Figure 8: The overall result of the chi-square compared with original result.

Because the purpose of the fitting is determination the minimum chi square, the determining of the optimized point isn't problem on the new calculation code whether the data fitting function was accurate.

THE GRAPHICAL INTERFACE

The graphical interface was implemented using qt-designer. The interface was created using a grid layout for the responsive interface when the size of the display is changed. The scan variables are created using pydm widgets in order to direct use epics access. Figure 9 shows the graphical interface.

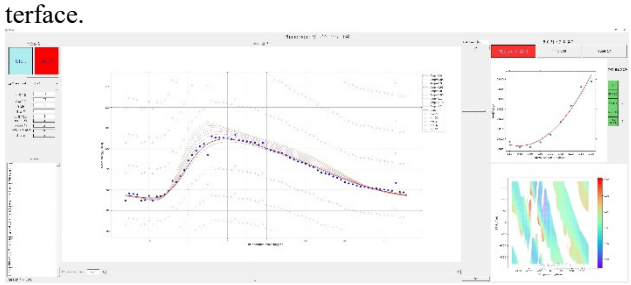


Figure 9: The graphical interface based on pydm.

CONCLUSION

The integrated application is implemented based on python language. The python is selected because of rapid development speed and high scalability. Rapid calculation process is replaced from the matlab function to python numba function and the scan algorithm is replaced from the java scan code to EPICS sscan record. Because the scan and the data analysis was integrated on the one application, the development or the maintenance is more flexibility and the interface is more user-friendly.

Furthermore, the future object of development is implement that's the auto scan function. If this function is implemented, the scan time will be significantly reduced.

REFERENCES

[1] Experimental Physics and Industrial Control System (EPICS), http://www.aps.anl.gov/epics

[2] Epics Channel Access for Python (PyEpics), https://cars9.uchicago.edu/software/python/pyepics3/

[3] numba, http://numba.pydata.org/

[4] Python Display Manager (PyDM), https://slaclab.github.io/pydm/

[5] The sscan record, https://epics.anl.gov/bcda/synApps/sscan/sscanDoc.html

[6] Han-Sung Kim, Hyeok-Jung Kwon, Seong-Gu Kim, Seok-Geun Lee, Young-Sub Cho, "RF Phase Scan for Beam Energy Measurement of KOMAC DTL", in *Proc. The Korean Nuclear Society Authumn Meeting Conf. (KNS'16)*, Gyeongju, Korea, Oct. 2016.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

THE AUTOMATIC LHC COLLIMATOR BEAM-BASED ALIGNMENT SOFTWARE PACKAGE

G. Azzopardi*, G. Valentino[2], B. Salvachua[1]
[1] CERN, Geneva, Switzerland,
[2] University of Malta, Malta

Abstract

The Large Hadron Collider (LHC) at CERN makes use of a complex collimation system to protect its sensitive equipment from unavoidable beam losses. The collimators are positioned around the beam respecting a strict transverse hierarchy. The position of each collimator is determined following a beam-based alignment technique which determines the required jaw settings for optimum performance. During the LHC Run 2 (2015-2018), a new automatic alignment software package was developed and used for collimator alignments throughout 2018. This paper discusses the usability and flexibility of this new package describing the implementation in detail, as well as the latest improvements and features in preparation for Run 3 starting in 2022. The automation has already successfully decreased the alignment time by 70% in 2018 and this paper explores how to further exploit this software package. Its implementation provides a solid foundation to automatically align any new collimation configurations in the future, as well as allows for further analysis and upgrades of its individual modules.

INTRODUCTION

The CERN Large Hadron Collider (LHC) is the world's largest particle accelerator, built to accelerate and collide two counter-rotating beams towards an unprecedented center-of-mass energy of 14 TeV [1, 2]. The LHC is susceptible to beam losses from normal and abnormal conditions, which can perturb the state of superconductivity of its magnets and potentially damage equipment. A robust collimation system handles beam losses of halo particles by safely disposing the losses in the collimation regions, with a 99.998% cleaning efficiency [3].

The collimation system consists of more than 100 collimators [4], each made up of two parallel absorbing blocks, referred to as jaws, inside a vacuum tank. The collimators are installed with a fixed rotational angle, depending on their location and functionality, which allows to clean in either the horizontal (H), vertical (V) or skew (S) plane. The jaws must be positioned symmetrically around the beam to ensure safe machine operation. Each jaw can be moved individually using two stepper motors at the jaw corners, allowing collimators to be positioned at different gaps and angles.

* gabriella.azzopardi@cern.ch

BEAM-BASED COLLIMATOR ALIGNMENT

Two types of beam instrumentation are available to align collimators; the Beam Position Monitoring (BPM) and the Beam Loss Monitoring (BLM) systems. BPM pick-up buttons are installed in 30% of the collimators, embedded in their jaws to provide a direct measurement of the beam orbit at the collimator location [5]. BPMs allow for a safe and fast alignment by analysing the electrode signals without needing to touch the beam. The remaining 70% of the collimators can only rely on dedicated BLMs positioned outside the beam vacuum, immediately downstream from the collimator [6]. BLMs are used to detect beam losses generated when halo particles impact the collimator jaws, such that characteristic spikes recorded in the losses indicate that the reference halo has been touched. The procedure of aligning collimators, with BLMs or BPMs, is referred to as beam-based alignment (BBA).

The beam-based alignment with BLM devices is performed via a four-step procedure established in [6]. This involves aligning a reference collimator in addition to the collimator in question (i). The reference collimator is taken to be the primary collimator (TCP) in the same plane (p) as collimator i. This creates a *reference halo* that extends into the aperture of collimator i. The procedure is to align the reference collimator before and after collimator i, as depicted in Fig. 1.

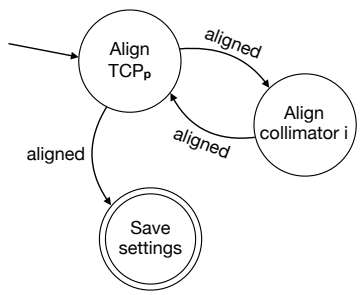


Figure 1: State machine of the beam-based alignment with reference collimator (TCP), from [7].

Once the beam has a well-defined halo amplitude shaped with the primary collimator, the alignment of a collimator can begin. A collimator is aligned by first aligning both jaws simultaneously towards the beam, followed by independently aligning the collimator jaws sequentially. The procedure, as depicted in Fig. 2, involves selecting a BLM threshold to stop the jaw movement when the recorded beam losses

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

surpass this threshold. Once the movement is stopped, the final step is to determine whether the jaw is aligned. A jaw is classified as aligned when an alignment spike pattern is detected in the losses, on two separate alignment instances. An alignment spike indicates that the collimator jaw has reached a transverse position closer to the beam than the reference collimator.

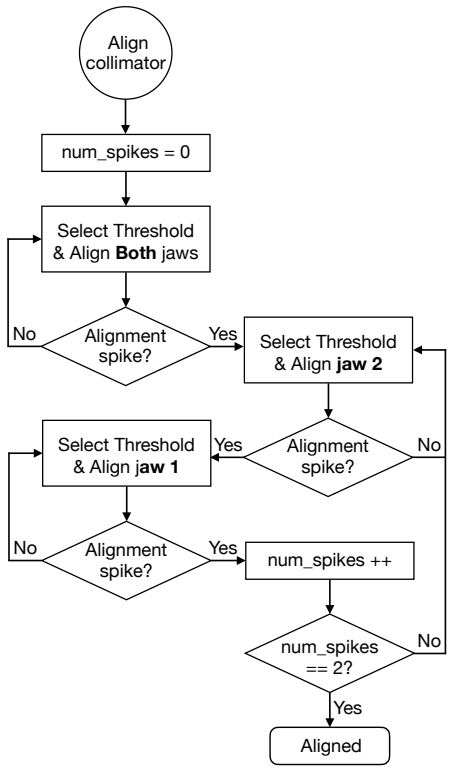


Figure 2: Flowchart of the beam-based alignment of an individual collimator, from [7].

The BBA procedure involves moving the collimator jaws towards the beam in steps of 5-20 μm whilst monitoring the beam loss signal recorded in the collimator's respective BLM. The alignment of any collimator relies on being able to identify an alignment spike when a jaw touches the reference halo: in this condition, the jaw reached the same aperture defined by the primary collimator with an accuracy of the step size. From this measurement, one can infer the local orbit position and the relative opening with respect to the primary collimator, which is used to establish the collimation hierarchy [8]. The BBA depends on being able to efficiently recognize alignment spikes and distinguish them from spurious signals that are often encountered, such that a collimator must continuously move towards the beam ignoring any spurious spikes, until a clear alignment spike is observed.

At the start of each year, and in case of major machine configuration changes during a run, the LHC goes through a commissioning phase to ensure that everything is correctly set up and ready for nominal operation. During the commissioning phase various alignment campaigns take place in order to set up the correct collimation hierarchy. Such campaigns make use of the BBA to align all collimators during their respective machine stages (injection, top energy, collisions, etc.).

To speed-up the procedure during such large alignment campaigns, the first step is to move all collimators in the same plane in parallel towards the beam, until the losses exceed the predefined thresholds. Once all collimators stop moving, each collimator is then aligned individually. The primary collimator of the plane is also aligned before and after moving the plane collimators in parallel to create the reference halo. The entire procedure is displayed in Fig. 3.

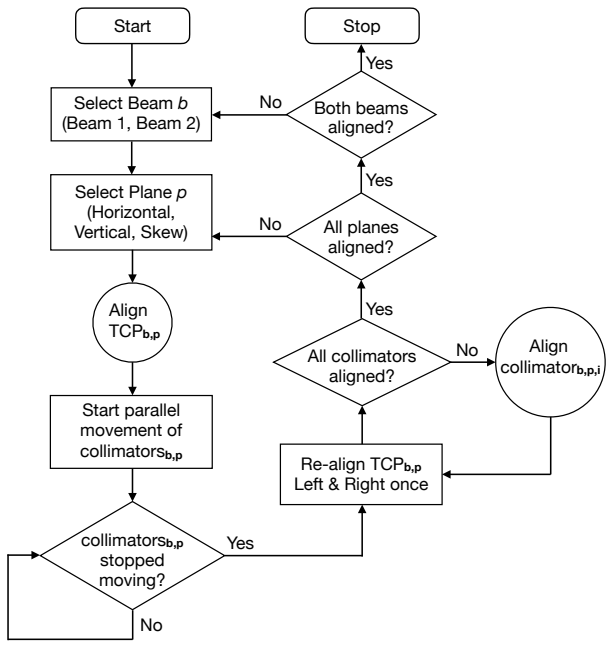


Figure 3: Flowchart of a collimator alignment campaign using the BBA in Fig. 2, from [7].

In addition to this, collimators from the two beams can be aligned in parallel to further speed-up the procedure. In such cases, one must consider the cross-talk across collimators, whereby losses generated by a collimator are not only detected by its corresponding BLM, but also by other BLM detectors around the LHC [9].

LHC COLLIMATION SOFTWARE ARCHITECTURE

Collimator alignments are performed from the CERN Control Center using a top-level application, allowing users to control collimators and monitor their BLM signal. The software architecture designed for the collimation system is implemented via a 3-tier structure as shown in Fig. 4.

The hardware consists of actuators, sensors and measurement devices. These allow for adjusting a number of parameters, including the collimators' jaw positions.

This hardware is abstracted and controlled in real-time through FESA (Front-End Software Architecture) [11], the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

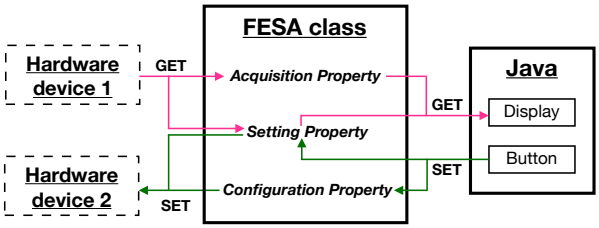


Figure 4: Software architecture diagram showing an example of FESA acting as the middleware between the Java application and hardware devices, from [10].

C/C++ framework used to develop LHC ring front-end equipment software which sends motor step commands [12]. FESA is a complete environment to design, develop, test and deploy real-time control software for front-end computers (FECs), standardizing front-end software development.

The high-level control system communicates with the FEC through devices. A device is the elementary control unit that exposes a public interface made up of properties. Most devices are software abstractions of the hardware (e.g. a collimator). FESA devices are grouped into a *FESA class* which defines the: property interface, private data and real-time behavior, for all devices belonging to that class.

Finally, the top level consists of Java Swing GUI applications. These interact with the FESA middleware framework through the Java API for Parameter Control (JAPC) [13], to control accelerator devices.

AUTOMATIC BBA IMPLEMENTATION

The fully-automatic alignment software introduced at the end of Run 2, was used throughout 2018 for all collimator alignments [14,15]. When provided with a list of collimators, this new tool is able to automatically align them following the procedure defined in Fig. 3.

The automation acts as a supervisor of the beam-based alignment, as it determines the order of tasks to be executed, controls the flow of collimator alignments, and reacts to the different states reached during an alignment campaign. The entire fully-automatic alignment is implemented on top of the beam-based alignment, within a dedicated FESA class - *CollAlignSupervisor* [10].

This software relies on the automation of three main components:

- Collimator Selection - The parallel alignment of collimators is automatically determined to minimize any cross-talk across the two beams. This makes use of offline analysis performed on 2018 commissioning data to determine the pairs of collimators that can be aligned in parallel without affecting each other [7]. The results of this analysis are available in an external file which can be updated in real-time for the FESA class to use directly.
- Threshold Selection - The collimator movement towards the beam is stopped when the BLM losses exceed this predefined threshold. The threshold is automatically selected and updated based on the real-time BLM losses detected at the collimator. Data from 2016 alignment campaigns was analysed to determine a suitable algorithm, which is implemented directly within the FESA class [16].
- Spike Classification - A "spike" is a signal triggered when the BLM losses reach the selected threshold. Supervised machine learning (ML) is used to automatically classify the BLM loss spikes into two classes; alignment spike or spurious spike. Features are extracted from the BLM loss signal and are used as inputs to the pre-trained ML models for classification [17]. The pre-trained models are available in external files which can be updated in real-time for the FESA class to use through Python. The FESA class handles the Python call for classification in a dedicated thread, which then communicates the final result to the original thread executing the fully-automatic alignment [10].

These three components are developed as individual modules within the automatic alignment software package, independently available for any improvements/upgrades, as demonstrated in [18].

Multi-threading

At any point in time only two collimators can be aligned in parallel, one from each beam, due to the same reference halo used by collimators in the same plane, and cross-talk restrictions across planes. In order to provide this functionality, two devices are defined for the *CollAlignSupervisor* FESA class, allowing two instances of the fully-automatic alignment software to run in parallel, i.e. two threads.

Each thread constantly communicates the following:

- The current beam and plane being aligned.
- The status of the primary collimator in the plane, i.e: moving and waiting statuses.
- The collimator ongoing alignment, for cross-talk purposes.
- The global wait status, i.e. if any thread is waiting for an action from the other thread.

GUI Application Communication

The *CollAlignSupervisor* FESA class provides a number of properties available to the GUI application for collimator control and monitoring. To begin a new alignment the user selects the list of collimators to be aligned, which are automatically sorted into the two beams and aligned sequentially. To align the two beams in parallel the user must open two instances of the GUI application, as this is common practice in the Control Centre. In this case, the user must select two separate lists of collimators to be aligned, one per beam.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 1: *CollAlignSupervisor* Properties Available for User Monitoring

Property	State	Definition
Auto status	ERROR_AUTO(-1)	Alignment halted with error.
	HALT_AUTO(0)	User paused/stopped alignment.
	PLAY_AUTO(1)	Alignment ongoing.
Align status	PARALLEL_ALIGN(-3)	Plane collimators moving in parallel.
	PARALLEL_DONE(-2)	Parallel plane movement ready.
	IGNORE(-1)	Property not in use.
	START_ALIGN(0)	Collimator alignment starting.
	DONE_ALIGN(1)	Collimator aligned.
	SAVE_SETTINGS(2)	Collimator aligned and settings saved.
Parallel status	DEADLOCK(-1)	Both beams waiting for each other.
	OK(0)	Property not in use.
	WAIT_CROSSTALK(1)	Wait due to crosstalk with alignment in other beam.
	WAIT_PARALLEL(2)	Wait parallel plane movement in other beam.
	WAIT_PAUSE(3)	Wait for other beam to pause alignment.
	WAIT_TCP(4)	Wait for TCP alignment in other beam.
	WAIT_CHANGE(5)	Wait for collimator to change in other beam.
Parallel message	-	Any message to display in GUI.
TCP status	TCP_NOT_DONE(-4)	TCP not yet aligned before collimator.
	TCP_DONE(-3)	TCP aligned before collimator.
	PARALLEL(-2)	TCP aligned before parallel plane movement.
	NOT_MOVING(-1)	Property not in use.
	BEFORE_COLL(0)	Ongoing TCP alignment before collimator.
	AFTER_COLL(1)	Ongoing TCP alignment after collimator.
Collimator status	-	Name of collimator ongoing alignment.
Jaw status	NO_JAW(-1)	Property not in use.
	FIRST(0)	First jaw alignment ongoing.
	SECOND(1)	Second jaw alignment ongoing.
	BOTH(2)	Both jaws alignment ongoing
Spike class	NO_CLASS(-2)	Property not in use.
	ERROR(-1)	Error on classification.
	NO_SPIKE(0)	BLM signal classified as non-alignment spike.
	SPIKE(1)	BLM signal classified as alignment spike.

Two FESA properties are used by the Java application to start new alignments:

- Multithreading status - FESA exposes which thread(s) are currently in use, to begin new alignments on available threads.
- Beam status - The beam(s) being aligned by each thread to ensure the user does not attempt to align collimators from the same beam in parallel. In cases of beam overlap the user is requested to wait until the ongoing alignment in the respective beam is completed or paused/stopped by the user.

Once an alignment campaign begins, the user is allowed to pause/resume or stop the alignment. The Java application keeps track of the subset of collimators which are still to be aligned, such that only these are sent to FESA. The user can monitor the status of the automatic alignment, as the FESA class constantly communicates the status through the properties listed in Table 1.

Automatic Alignment Features

The automatic alignment was designed to be autonomous. It must independently "make decisions" in real-time based on the alignment status, until the alignment of all selected collimators is complete. In addition to this, a number of "smart" features have been introduced to the automatic alignment when aligning the two beams in parallel. The aim of these features is to imitate, as much as possible, the decisions a user would take when aligning collimators. This software must align the collimators as efficiently as possible, whilst ensuring the correct alignment and must avoid classifying a

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

collimator "aligned", if it has not reached the reference halo. The incorporated "smart" features include:

- Alternating between collimators from the two beams that must be aligned sequentially due to cross-talk.
- Automatically wait when the collimators in the other beam are performing the parallel plane movement.
- Before starting the parallel plane movement, wait for the collimator in the other beam to finish its alignment (with TCP) and save settings.
- If both beams are close to start the parallel plane movement, wait to move both planes together.
- On pause then resume, the collimators in the current plane are assumed to have already moved in parallel.
- On pause/stop, the other thread will automatically resume (if waiting), as expected.
- Any deadlocks are handled by resuming the alignment on thread 1.

GUI APPLICATION USABILITY

The users have various options newly available at the start of Run 3, including:

- After selecting the list of collimators to align, subsets of the list can be further selected for grouping alignments.
- Collimators can be manually removed from the list during the alignment and re-added at a later stage, in cases of issues with particular collimator or if different settings are to be used for different collimators.
- Preset selections for aligning subsets of collimators, e.g. aligning only collimators with/without BPMs.
- Aligning the TCP before/after collimators (recall Fig. 1) is now optional for the user to select. A combination of having a subset of collimators aligned with TCP and others without TCP is also available.
- Moving the collimators in the current plane in parallel is now optional. (By default, this is not selected at flat top due to a beam dump that occured in the past [7].)

Overall the user always has full control of the automatic alignment with the play/pause/stop buttons. To select subsets of collimators or to change any settings, the user must first pause or stop the alignment. Finally, closing the application is an automatic stop if any alignment is ongoing.

ALIGNMENT OUTLOOK

Introducing the automatic alignment software has made various alignment configurations accessible and feasible to execute on a regular basis:

- Aligning collimators at an angle potentially allows for tighter collimator settings [19]. However, this procedure is longer, as the BBA is applied at different jaw angles to find the most optimal one. The automatic software makes such angular alignments more accessible.
- Any combinations of collimators, with or without TCP, are now available for the users to perform more efficiently with minimal effort.
- Specific collimator alignments can be performed more frequently during LHC operation, rather than having to wait for dedicated beam time for alignment campaigns.
- Collimator configurations, e.g. for ion beams, can now become more independent. In this case, the automatic alignment allows for evaluating dedicated configurations for ion beams, rather than being bound to keeping the identical setup used for proton beams.

CONCLUSION

The LHC is equipped with more than 100 collimators to protect its sensitive equipment. In order to ensure safe beam operation, dedicated collimator alignment campaigns are performed at the start of each year and whenever the LHC configuration is changed, to determine the collimator settings for nominal operation. This is a critical phase to ensure safe operations and various improvements were carried out year after year, in order to reduce the required time while ensuring the accurate alignment necessary.

The majority of collimator alignments are applied using the BBA procedure relying on dedicated beam loss monitors. This procedure has been fully-automated by means of a new software package introduced in 2018, providing a solid foundation to align collimators automatically, yielding a 70% speed-up in collimator alignments [14, 15].

The automatic alignment software package relies on three components that have been developed as independent modules; collimator selection, threshold selection and spike classification. This implementation design provides the flexibility to independently extract each of these modules for further analysis and possibly replace with software upgrades.

This software package is readily available with various new features, to be used as the primary tool for collimator alignments during Run 3. This will enable the configuration of new collimator settings for different machine setups, that were not feasible to be determined in the past due to time restrictions. Moreover, this will make collimator alignments accessible during LHC operation to possibly align subsets of collimators more frequently, rather than waiting for large alignment campaigns to be scheduled.

ACKNOWLEDGEMENTS

The authors would like to thank the collimation team (BE-ABP-NDC) for their feedback as front-end users and the LHC operational crew (BE-OP) for support during software tests with beam.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] L. Evans, "The Large Hadron Collider", in *New J. Phys.*, vol. 9, no. 9, 2007.

[2] L. Evans and P. Bryant, "LHC machine", in *J. Instrum.*, vol. 3, no. 8, 2008.

[3] R. W. Assmann *et al.*, "Requirements for the LHC collimation system", in *Proceedings of the European Particle Accelerator Conference*, Paris, France, pp. 197 – 199, 2002.

[4] G. Azzopardi, *et al.*, "LHC Collimation Controls System for Run III Operation", in the 18th International Conference on Accelerator and Large Experimental Physics Control Systems, Shanghai, China, 2021, paper THPV012, this conference.

[5] G. Valentino, *et al.*, "Final implementation, commissioning, and performance of embedded collimator beam position monitors in the Large Hadron Collider", *Phys. Rev. Accel. Beams*, vol. 20, no. 8, p. 081002, 2017. doi:10.1103/PhysRevAccelBeams.20.081002

[6] R. W. Aßmann, *et al.*, "Expected performance and beam-based optimization of the LHC collimation system", in *Proceedings of the European Particle Accelerator Conference*, Lucerne, Switzerland, 2004, pp. 1825–1827.

[7] G. Azzopardi, "Automation of the LHC Collimator Beam-Based Alignment Procedure for Nominal Operation", Ph.D. thesis, Univeristy of Malta, 2019.

[8] R. Bruce, *et al.*, "Calculations of safe collimator settings and β^* at the CERN Large Hadron Collider", *Phys. Rev. Accel. Beams*, vol. 18, no. 6, p. 061001, 2015. doi:10.1103/PhysRevSTAB.18.061001

[9] G. Azzopardi, B. M. Salvachua Ferrando, and G. Valentino, "Data-driven cross-talk modeling of beam losses in LHC collimators", in *Physical Review Accelerators and Beams*, vol. 22, no. 8, p. 083002, 2019. doi:10.1103/PhysRevAccelBeams.22.083002

[10] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Software Architecture for Automatic LHC Collimator Alignment Using Machine Learning", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 78–85. doi:10.18429/JACoW-ICALEPCS2019-MOCPL04

[11] A. Guerrero, J. J. Gras, J-L. Nougaret, M. Ludwig, M. Arruat, and S. Jackson, "CERN Front-End Software Architecture for Accelerator Controls", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Gyeongju, Korea, 2003, paper WE612, pp. 342–344.

[12] A. Masi, R. Losito, and S. Redaelli, "Measured Performance of the LHC Collimators Low Level Control System", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Kobe, Japan, 2009, paper WED001, pp. 612–614.

[13] V. Baggiolini, *et al.*, "JAPC - the Java API for parameter control (designing for smooth evolution)", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Geneva, Switzerland, 2005.

[14] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Operational Results of LHC Collimator Alignment Using Machine Learning", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 1208–1211, doi:10.18429/JACoW-IPAC2019-TUZZPLM1

[15] G. Azzopardi, B. M. Salvachua Ferrando, G. Valentino, and S. Redaelli, "Operational Results on the Fully-Automatic LHC Collimator Alignment", *Physical Review Accelerators and Beams*, vol. 22, no. 9, pp. 093001, 2019. doi:10.1103/PhysRevAccelBeams.22.093001

[16] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Automatic Beam Loss Threshold Selection for LHC Collimator Alignment", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 208–213. doi:10.18429/JACoW-ICALEPCS2019-MOPHA010

[17] G. Azzopardi, G. Valentino, A. Muscat, and B. M. Salvachua Ferrando, "Automatic spike detection in beam loss signals for LHC collimator alignment", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 934, pp. 10–18, 2019. doi:10.1016/j.nima.2019.04.057

[18] G. Azzopardi *et al.*, "New Machine Learning Model Application for the Automatic LHC Collimator Beam-Based Alignment", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, China, 2021, paper THPV040, this conference.

[19] G. Azzopardi, A. Mereghetti, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Automatic Angular Alignment of LHC Collimators", in *Proc. ICALEPCS2017*, Barcelona, Spain, Oct. 2017, paper TUPHA204. doi:10.18429/JACoW-ICALEPCS2017-TUPHA204

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

THE LINAC4 SOURCE AUTOPILOT

M. Hrabia, M. Peryt, R. Scrivens, CERN, Geneva, Switzerland
D. Noll, European Spallation Source, Lund, Sweden

Abstract

The Linac4 source is a 2MHz, RF driven, H- ion source, using caesium injection to enhance H- production and lower the electron to H- ratio. The source operates with 800µs long pulses at 1.2 second intervals. The stability of the beam intensity from the source requires adjustment of parameters like RF power used for plasma heating.

The Linac4 Source Autopilot improves the stability and uptime of the source, by using high-level automation to monitor and control Device parameters of the source, in a time range of minutes to days.

This paper describes the Autopilot Framework, which incorporates standard CERN accelerator Controls infrastructure, and enables users to add domain specific code for their needs. User code typically runs continuously, adapting Device settings based on acquisitions. Typical use cases are slow feedback systems and procedure automation (e.g. resetting equipment).

The novelty of the Autopilot is the successful integration of the Controls software based predominantly on Java technologies, with domain specific user code written in Python. This allows users to leverage a robust Controls infrastructure, with minimal effort, using the agility of the Python ecosystem.

INTRODUCTION

The CERN Linac4 ion source is a 2MHz RF driven H^- ion source, using caesium injection to enhance H^- production and lower the electron to H^- ratio. The source operates for Linac4 with 800µs long pulses at 1.2 second intervals.

The stability of the H^- beam intensity from the source over the period of minutes to days requires adjustment of parameters like the RF power used for plasma heating. Controlling the source on this timescale is the objective of the Autopilot. It is not conceived to work from pulse to pulse, or within a pulse, which would require dedicated systems at the front-end computer level.

The Autopilot Framework was developed in 2019 and used successfully to automatically tune the source during the test runs for Linac4. It is currently operating 24/7 helping the operations team to run the linear accelerator. Another instance has been successfully deployed in the CERN ion linac, Linac3. It replaces a highly specific version developed in 2017 [1], which lacked flexibility.

FRAMEWORK

The Autopilot Framework, henceforth referred to as the framework, is a set of services and tools (see Fig. 1.) that allow the users to deploy and execute the algorithmic tasks in a self-service manner, where the framework provides the necessary tooling and infrastructure. Typical use cases are slow feedback systems and automated procedures (like resetting and restarting equipment that has stopped due to a fault state).

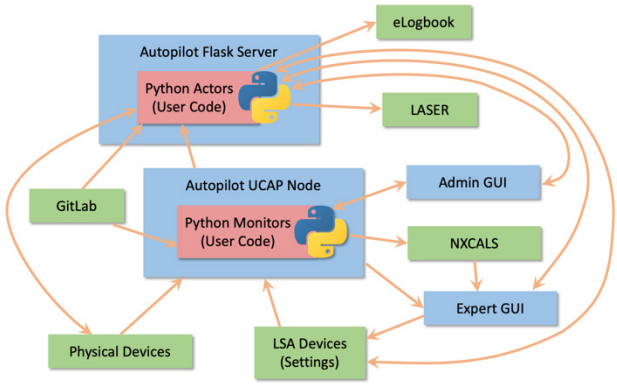


Figure1: Schematic diagram of the Autopilot Framework. Blue boxes signify the Autopilot framework components.

The framework allows the users to subscribe to the Control parameters of their choice, process the received values, and publish the output as properties of Virtual Devices, i.e. Control Devices that are implemented exclusively in software, with no hardware component. Those Virtual Devices can in turn be used like any other Control Device, in particular they can provide inputs to the user-supplied regulation algorithms (subsequently referred to as actors) that interact with the Linac4 H^- ion source.

A particularity of the framework is that although it is based on the accelerator Controls software stack, that is predominantly developed in Java, it allows the user code to be written in Python. Another characteristic of the framework is that it leverages the well-established services and components that form the accelerator Controls, in particular the Controls Configuration Service (CCS) [2], Controls Middleware (CMW) [3], Role-Based Access System (RBAC) [4], the Unified Controls Acquisition and Processing (UCAP) framework [5], LSA (Accelerator Settings Management) [6], LASER (Accelerator Alarms Service) [7], and NXCALS (Accelerator Logging Service) [8], thus reducing the need for developing custom components.

At the core of the framework lies a UCAP node that subscribes to physical and virtual Devices according to the recipes provided by the users. The recipes also contain the triggering conditions. When those conditions occur, the events containing the accumulated data are constructed and forwarded to the user-provided transformation routines, known as “Monitors”. The Monitors calculate the output values and give them back to UCAP, for publishing as virtual Device Properties through controls middleware.

Along with the UCAP recipes, and the Monitors, the Flask server also manages the user-provided control tasks called “Actors”. These are basically Python scripts whose job is to act upon the updates received from the Monitors

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

and perform the necessary actions to keep the parameters of the Linac4 source within the requested limits. The benefit of the Actors using Monitors, is that the UCAP system takes care of synchronizing multiple data inputs, allowing the Actor to receive all required data in a single subscription.

Server

The server is a REST service powered by the Flask micro-framework combined with Gunicorn as HTTP Server. The REST communication is hidden behind a simple web interface, which serves as a bridge between the user and the framework itself. The service was built with the use of Acc-Py, a CERN Accelerator Controls Python distribution.

The web interface consists of three subpages, each dedicated to a separate part of the Autopilot Tasks: Monitors, Actors and UCAP configuration. Each of them provides a status view of uploaded user scripts, as well as interactive tools to manage the content of the service, such as adding and removing script files or starting and stopping Task execution.

To make sure that the communication is secure, an HTTPS connection to the server has been set up, with the help of tools provided by the CERN Certification Authority, giving an encrypted communication channel between the service and the user.

To restrict access to authorised users, Role Based Access (RBAC) is used for user authentication. On top of that, to distinguish between the different types of users and restrict access even further, three different internal roles (Guest, Operator, SuperUser) have been defined allowing for the fine-tuned levels of control over the framework.

The web interface also provides a set of features to help the users service their Tasks in case they do not behave as expected, such as:

- Checking python script correctness at start-up time – when the user tries to start a new python script, the check for basic code errors, like wrong indentation level, is performed. Since obvious mistakes like this would prevent the script from running correctly, an error message specifying the reason of the failure is shown to the user, and the new process is never started.
- Logging mechanisms – the user is provided with a predefined logger, which can be used inside user's script to log any useful information during code execution. User logs are stored on the server and are easily accessible through the web-interface.
- Automatically catching runtime exceptions – execution errors which may happen at runtime and are not handled by user code are captured by the framework and put into the user logs for post-mortem analysis.
- Subscription status preview – for each Monitor Task, the list of all its Data Sources and their connection statuses is provided. This allows the user to see in real time if the data from each Data Source is available. For quick debugging of broken subscriptions, a remote 'test get' feature is provided to query the input devices.
- File peeking – the user can visualise the task code directly in the web interface.
- Test data generation – before the Monitor script will be uploaded to the server it has to be written offline. Python at its core is a language where it is easy to make coding errors, so we give the possibility to record a specified amount of input data from UCAP in the form of JSON files, that can be downloaded by the users and fed to their algorithms offline, making it easier to find some obvious problems upfront, before the final version will be uploaded to the server.

Source Code Management – GitLab

To manage the state and content of user script files, the server is connected to a dedicated git repository hosted on CERN's GitLab service. The repository contains a configuration file listing explicitly all Actors, Monitors and UCAP configuration files. This allows the server to easily recognize the expected role of a given file and handle it accordingly.

By using GitLab, the history of changes is available for all files, providing a clear trace of how content was changed, by whom and when. It also gives a simple way to roll back to a previous version of the file in case the new version is not behaving as expected.

The server uses the GitLab REST API to compare files from the repository with those currently hosted on the server as well as download them if necessary.

From the server perspective, updating any script file is as simple as two clicks on the web panel. First click to 'check' for any changes, where the server compares commit IDs of the files, and the second click to 'synchronize' the file, meaning downloading the latest version of the file from the GitLab repository if it is newer than the version currently on the server.

For safety reasons the script files can only be synchronized if the corresponding Autopilot Task is not in a 'running' state. However, the version check can be done at any time, as it does not impact the content of the server.

When any script file has been recognized as outdated – a notification label is shown next to the file name on the web panel, making it easily noticeable. As an added value, the label itself is a hyperlink to the GitLab webpage commit details, which highlights changes in the script file content between versions, thus allowing for a quick verification before deciding to synchronize script files to the server.

The server is also capable of being notified by a GitLab web hook automatically, after any file has been changed, sparing the user the necessity of invoking the 'check' operation manually.

No automation is foreseen for the non-interactive synchronization of script files. Since those files are actually running on the server, it is critical for stable operations, to only update runtime content when there is a safe operational window for doing that. This update can only be performed by super-users, as defined in the preceding section.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

UCAP

The Unified Controls Acquisition and Processing (UCAP) project aims to provide "a generic, self-service, controls data processing platform".

In simpler terms, UCAP is a platform that allows the creation of a standard Virtual Device, which receives data from a predefined list of signals, gives that data to the user-written or configured code (known as a Transformation), which can in turn perform any kind of calculations and processing, and publishes the result back to the Controls system using a standard middleware protocol. State is retained between calls to the user transformation, so that it is possible to apply transformations over multiple beam cycles (for example averaging a value over some time window).

At the core of the Linac4 Source Autopilot, Monitors are in fact UCAP Virtual Devices, publishing transformed data that is in-turn used by the Actors to interact with the Source. Corresponding data is stored in the logging database (NXCALS) and can be used later for purposes such as diagnostics.

Monitors

As explained above, Monitors are Virtual Devices running on a UCAP node. The Autopilot server simply acts as a man-in-the-middle between the user and the UCAP framework, meaning the user uploads UCAP configuration and transformation code files using the aforementioned web panel, and the Autopilot servers takes care of turning that into a request understandable by the UCAP framework.

It is the responsibility of the UCAP framework to create a new process to receive, transform and publish data. The Autopilot server does nothing more than ask UCAP to add/remove or start/stop the Virtual Device in question.

The data transformation code is written in Python using the provided ucap-python package, which contains all interfaces representing the data model of UCAP. The usage of this package is mandatory for every Autopilot Monitor script.

By design, Monitors only read and transform the data published by other Devices and do not interact themselves with the Control system.

Monitors can also receive configuration parameters for their execution by including LSA virtual settings values in their data input. For example, a minimum and maximum needed to perform a value validation can be created as a LSA virtual setting, and input to an Autopilot Monitor along with the real Device acquisition for comparison. In this way the LSA parameters allow many different ways to incorporate settings into Autopilot applications, at the same time keeping a history of changes to these settings inside the LSA settings history repository.

Actors

Actors are Tasks that continuously receive data produced by Monitors, and possibly some other sources such as LSA settings (both from real Devices, and Virtual Devices). Based on these inputs they can decide to invoke a particular action if necessary. Each Actor Task runs as a separate process.

Actor Tasks can use a special API to notify the Autopilot server about certain events by invoking specific actions, which are typically reflected on the web panels (and through other services like the electronic Logbook (eLogbook) and LASER alarms system). They give useful real-time feedback of a running Actor's behaviour. Invokable actions include:

- heartbeat – an Actor script can call this method to announce that it is alive. It is usually invoked by the user whenever new data is received by the Actor.
- set_last_acted – an Actor script can notify that it is performing an action it considers as an 'act' method (like applying a new hardware setting) and provide a short message describing it.
- set_last_error / clear_last_error – an Actor script can notify the server that it recognized a situation considered as an error, for example that it received corrupted data.
- abort_task – an Actor script can ask the server to cleanly abort the Task execution, for example where a set of conditions have no path defined, and the situation is considered unsafe for the Actor.
- write_to_elogbook – an Actor script can write a short message to the eLogbook service.
- set_alarm / clear_alarm – an Actor script can create a new alarm that will be shown on the LASER console expanding further possibilities of notifying operators that something on the Autopilot requires attention.

Unlike Monitors whose lifecycle is managed within UCAP, the lifecycle of Actor Tasks, is managed directly by the Autopilot server. This means the Autopilot server is responsible for creating a new process and shutting it down later when the user requests the Task to be started or stopped, respectively.

Graphical User Interface

The Autopilot can be monitored and, to a certain extent, controlled through an interactive expert application (GUI) that can be started from the Common Console Manager (CCM). The main screen of the GUI is the Beam Current Transformer (BCT) Stabilize panel. This panel provides a graphical overview of the current and historic state of the source current stabilization Task, discussed in some detail further in this document. It also allows one to start and stop the regulation Tasks, and to control the current set points.

Along with the BCT Stabilize panel, the GUI provides a number of additional panels with a similar functionality, namely the electron to H^- (eH) Ratio panel, the Caesium Flow panel, and the High Voltage (HV) Reset panel.

Integration with Accelerator Controls Services

The configuration of the input and output devices of the Autopilot is handled by the Controls Configuration Service.

To allow the Autopilot server to communicate its important actions to the outside world when necessary, Actor

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Tasks have the possibility to write messages to the eLogbook. Behind the scenes the server uses the REST API of the eLogbook to write messages from the Actor Task to the appropriate section of the logbook.

This functionality is limited solely to Actor Tasks, as they are the only processes whose actions can have implications for the operation of the accelerator, and thus should be announced to operators when necessary.

The Autopilot server is also integrated with the LASER (Alarms) service, giving Actor Tasks the possibility to create a new alarm that will be visible by operators on the LASER console. The functionality is limited to Actor Tasks for the same reason as explained above for the eLogbook integration.

LSA (Settings Management Service) is used by the Autopilot to manage Actor settings. This allows the standard Accelerator controls graphical applications to visualise data published by the Autopilot, and to influence the behaviour of the Autopilot.

NXCALS (Logging Service) is used by the Autopilot to log the values published by the Monitors. Data is also extracted from NXCALS when starting the Autopilot GUI, to prefill charts with a certain amount of historical data.

Deployment

The two core parts of the framework are the Autopilot Flask Server, and the Autopilot UCAP Node (see Fig. 2.). The Autopilot Flask Server, as an Acc-Py-based service, uses Acc-Py tools for the releases of new versions. However, for the deployment on its dedicated server, a custom deployment script had to be developed, as currently Acc-Py lacks tools for services deployment. The UCAP Node, being a Java application at its core, uses CBNG [9] for both releasing and deployment.

Both services run on the same server running CentOS 7.

USER TASKS FOR THE SOURCE AUTOPILOT

In this section we describe the user tasks available and their status as of this writing.

BCT Stabilize

This task is the main workhorse of the Autopilot and its effect is illustrated in Fig. 3. Its objective is to keep the H^- beam current constant by adjusting the forward power of the 2MHz RF used for plasma production and heating based on readings from a Beam Current Transformer (BCT) in the Low Energy Beam Transport line (LEBT).

Phase Stabilize

As well as the RF power to the ion source, the phase of the RF is also measured. This phase can be used to assess the efficiency of the coupling of the RFQ power to the source plasma. The Phase Stabilize Task monitors the measured phase over defined windows and compares them to the set point. If outside a dead band, the frequency is adjusted along the pulse to return to the phase set point.

HV Reset

Ion sources typically suffer from sparking events in the order of a few per day. The source power converters go into a fault state when an over-current is detected, and do not deliver the voltage until reset.

The HV Reset Monitor delivers the present status of the HV converters, as well as counting the number of times that they have gone from ON to FAULT status in the last 24 hours.

The Actor reads the repeated HV converter statuses from the Monitor, and when it moves from ON to FAULT status, it initiates a procedure to restart them.

eH Ratio

The eH ratio is a Monitor that uses the currents measured on the source high voltage power converters, and the BCT

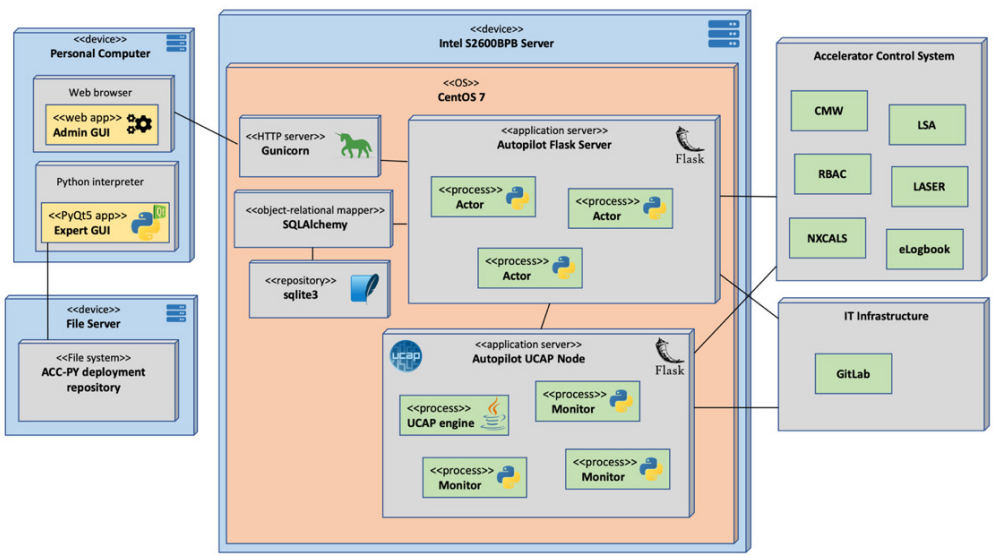


Figure 2: Autopilot deployment overview.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

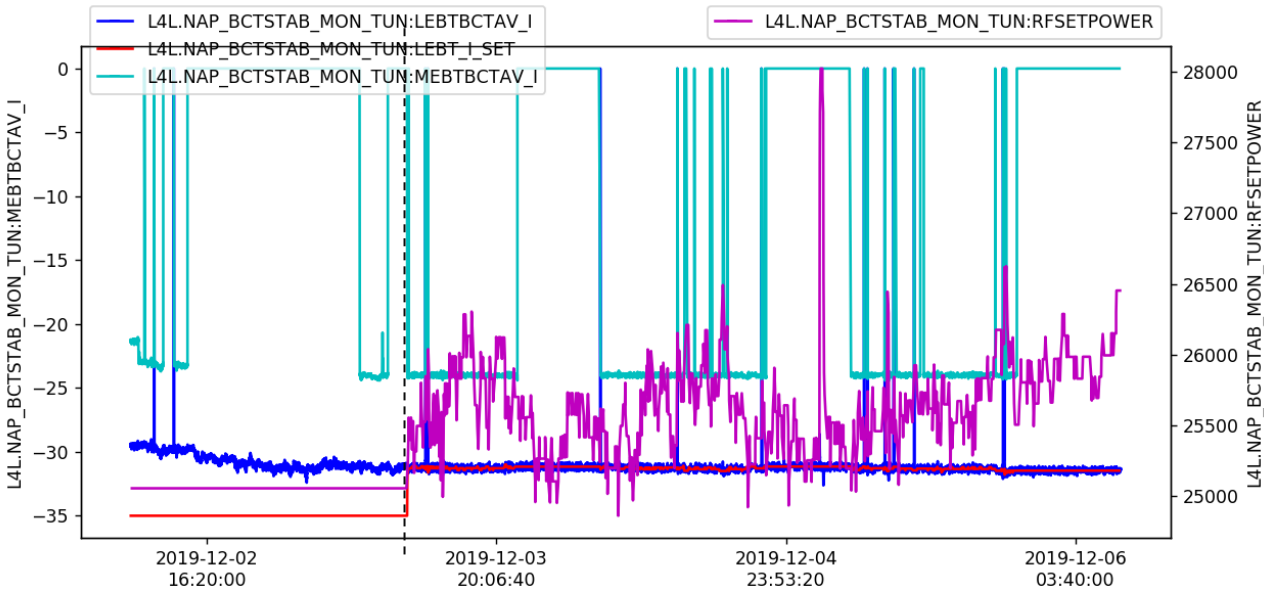


Figure 3: Beam intensity and RF set power from 2 December 2019 to 5 December 2019. Black vertical line shows the time after which the autopilot was started to maintain the MEBT beam intensity constant. Cyan= MEBT measured intensity. Blue=LEBT measured intensity, Orange=LEBT requested intensity, Purple=RF requested power.

current in the LEBT, in order to calculate the electron to H^- ratio from the source.

Cs Flow

The Linac4 ion source uses caesium to reduce the work function of the plasma electrode, which enhances negative ion production. Caesium is evaporated into the source, under vacuum, from a heated oven.

The amount of caesium that exits the oven can be estimated by scaling of the vapour pressure in the oven to a calibration point. Integrating the flow over time gives a total mass. This monitor produces this calculation in real-time, and allows the instantaneous flow and total mass data to be logged over time using the NXCALS system.

SUMMARY

Within the Linac4 Source Autopilot project, a framework has been developed to allow users to develop code in Python that runs on a stable server, and interfaces to several accelerator controls components to allow monitoring, settings, diagnostics, and testing. The framework is in use at CERN helping to control the Linac3 and Linac4 sources. The framework described in this paper can be deployed in various operational scenarios involving the integration of user code with general controls services. Although the accelerator Controls services leveraged by the framework are CERN specific, we are confident that fundamental building blocks of the framework can be reused outside the Organization.

Within this framework, tasks have been developed to continuously adjust the source RF power and perform automatic resets of the High Voltage converters. Thanks to this the stability of the source has improved remarkably over this period.

ACKNOWLEDGEMENTS

The authors would like to thank the Linac4 ion source team in BE-ABP for providing the motivation for this project and testing time, as well as to the software teams within BE-CCS for their work to make this project possible, in particular the UCAP, Acc-Py, and LASER teams.

REFERENCES

[1] G. Voulgarakis, J. Lettry, S. Mattei, B. Lefort, and V. J. Correia Costa , "Autopilot regulation for the Linac4 H− ion source", *AIP Conference Proceedings* 1869, 030012 (2017), doi: 10.1063/1.4995732

[2] B. Urbaniec and L. Burdzanowski, “CERN Controls Configuration Service - Event-Based Processing of Controls Changes”, presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV043, this conference.

[3] J. Lauener and W. Sliwinski, “How to Design & Implement a Modern Communication Middleware Based on ZeroMQ”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 45-51. doi:10.18429/JACoW-ICALEPCS2017-MOBPL05

[4] W. Sliwinski, P. Charrue, and I. Yastrebov, “Status of the RBAC Infrastructure and Lessons Learnt from its Deployment in LHC”, in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper WEMMU009, pp. 702-705.

[5] L. Cseppento and M. Buttner, “UCAP: A Framework for Accelerator Controls Data Processing @ CERN”, presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV039, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV018

[6] D. Jacquet, R. Gorbonosov, and G. Kruk, "LSA - the High Level Application Software of the LHC - and Its Performance During the First Three Years of Operation", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper THPPC058, pp. 1201-1204.

[7] Z. Zaharieva and M. Buttner, "CERN Alarms Data Management: State & Improvements", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper MOPKN011, pp. 110-113.

[8] J. P. Wozniak and C. Roderick, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA163.
doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[9] L. Cseppento, V. Baggiolini, E. Fejes, Zs. Kovari, and N. Stapley, "CBNG - The New Build Tool Used to Build Millions of Lines of Java Code at CERN", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 789-793.
doi:10.18429/JACoW-ICALEPCS2017-TUPHA163

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

RENOVATION OF THE BEAM-BASED FEEDBACK CONTROLLER IN THE LHC

L. Grech*, G. Valentino, University of Malta, MSD 2080 Msida, Malta
D. Alves, A.Calia, M. Hostettler, J. Wenninger, S. Jackson, CERN, 1211 Geneva 23, Switzerland

Abstract

This work presents an extensive overview of the design choices and implementation of the Beam-Based Feedback System (BBFS) used in operation until the LHC Run 2. The main limitations of the BBFS are listed and a new design called BFCLHC, which uses the CERN Front-End Software Architecture (FESA), framework is proposed. The main implementation details and new features which improve upon the usability of the new design are then emphasised. Finally, a hardware agnostic testing framework developed by the LHC operations section is introduced.

INTRODUCTION

The Large Hadron Collider (LHC) was designed to handle particle momenta between one and two orders of magnitude higher than previous accelerators [1]. The highly energetic beams inside the vacuum chambers are stripped off of halo particles by a beam cleaning and machine protection system, also known as the collimation system. The collimation system required the machine tolerances to be tightened to work effectively. As a result, the LHC was the first proton accelerator to require automatic feedback control systems on various beam and machine parameters [2].

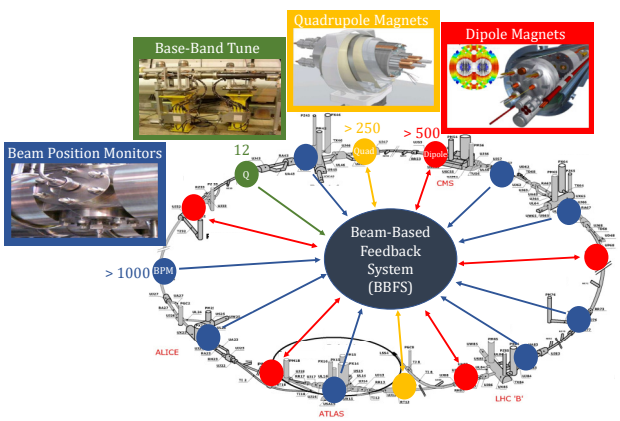


Figure 1: Schematic view of the BBFS in the LHC complex.

In this work, Beam-Based Feedback System (BBFS) denotes the program which was originally responsible for implementing the feedback control of the Radio Frequency (RF) systems, orbit and tune [3]. Figure 1 shows the BBFS within the LHC complex. The systematic beam energy offset from the ideal trajectory can be inferred from the orbit and in turn, the orbit is obtained from the measurements of 1088 Beam Position Monitors (BPMs) placed at different locations in the LHC [2, 4]. The Base-Band Tune (BBQ) systems are responsible for estimating the horizontal and vertical tunes of both beams in the LHC [5]. It is well known that the tune estimates from the BBQ were unstable due to 50 Hz noise harmonics present in the BBQ spectra [6, 7]. This instability also caused the Tune Feedback (QFB) to frequently switch off. As a consequence, the QFB was used intermittently and only when necessary, e.g. at the start of the ramp.

In its original design, the BBFS was foreseen to automatically control the coupling and chromaticity as well. Both these quantities are derived estimates from the BBQ system measurements. Considering that the tune estimates were often unstable, the control of coupling and chromaticity was deemed impractical to be used in operation, despite being implemented in the code.

The BBFS was made up of two components, which were historically named the Orbit Feedback Controller (OFC) and the Orbit Feedback Service Unit (OFSU). The OFC comprised the main program written in C++ and was primarily responsible for communicating with real-time Front-End Computers (FECs) to obtain beam measurements and applying magnetic corrections. The OFSU was implemented using the Front-End Software Architecture (FESA) framework used at CERN [8].

This work will provide a summary of the design and the limitations of the BBFS which was used in operation until the end of Run 2 in 2018. The BBFS underwent renovation during the LHC Long Shutdown 2 (LS2) and the upgraded version is called the Beam Feedback Controller LHC (BFCLHC). The BFCLHC is a FESA-based application which incorporates all the useful functionality of the BBFS, along with new features requested by the LHC operators. During LS2, our colleagues from the LHC operations section also developed a testing framework for the BFCLHC which for the first time allows closed loop tests to be performed on the feedbacks offline.

DESIGN UNTIL LHC RUN 2

Figure 2 illustrates a more detailed view of the BBFS architecture which is comprised of the OFSU and the OFC. The OFSU is connected to the OFC via a private Ethernet connection where Transmission Control Protocol (TCP) is used for lossless communication of the OFC settings and User Datagram Protocol (UDP) is used to stream real-time acquisition BPM, BBQ and magnet data from the OFC to the OFSU. Measurement data coming from the BPM and the BBQ systems is received by the OFC in the form of User

* leander.grech.14@um.edu.mt

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Datagram Protocol (UDP) packets. The BPM packets arrive from 67 FECs, which can interface up to 18 BPMs each. In addition, longitudinally consecutive BPMs are interleaved amongst at least two, geographically nearby FECs, so that if a FEC fails, a contiguous section worth of measurements is not entirely lost. Similarly, the calculated current corrections for the magnets are sent to the Power Converter (PC) gateways via UDP packets. These gateways are connected to the Function Generator/Controllers (FGCs) which control the current flowing in the magnets [9].

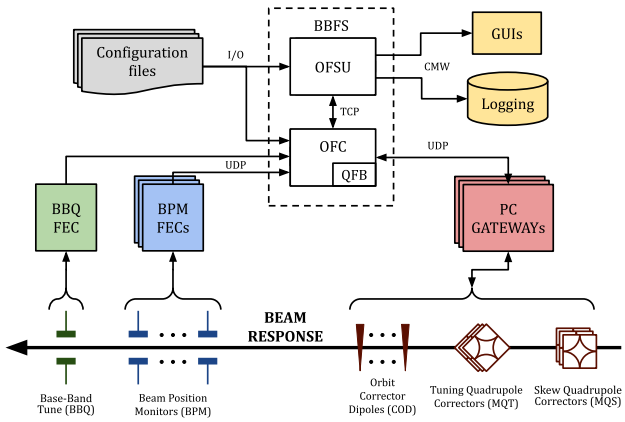


Figure 2: Schematic view of the data paths and a more detailed view of the BBFS architecture.

The principle design objective of the BBFS was to obtain a deterministic behaviour of the feedbacks on the hardware available at the time. The modular nature of the BBFS allowed the OFC to collect information about the tune and orbit in real-time, and calculate the required magnet corrections at a rate of 25 Hz. By keeping the program structure of the OFC simple, the designers could ensure a deterministic execution [2].

The OFC was written in object-oriented C++ and relied heavily on ROOT libraries. ROOT is written and maintained by CERN and was originally developed for efficient calculations on very large data accumulated by high-energy physics experiments [10]. The programming paradigm of ROOT is object-oriented where all objects derive from what is called a TObject. All TObjects can use the various functionalities implemented in ROOT, such as I/O handling, memory management and error handling. The OFC consists of several TObject-derived classes, which implement all the required functionality. The core of the OFC was a loop-based program, where every iteration was one simultaneous pass through the control loops of the OFC and the QFB. The latest version of the OFC used ROOT Release 5.34/20 [11].

Figure 2 also shows that the OFSU served as an interface for the OFC and was used by the LHC operators to monitor and control the OFC [3]. The OFSU was also responsible for loading some of the configuration files, which contained the settings and beam optics required by the OFC. This information was communicated to the OFC upon initialisation as well as by request of the operators. The OFSU also served as a hub for the data collected by the OFC and as a result a myriad of Graphical User Interfaces (GUIs) used by the LHC operators depended on the data relayed by the OFSU. One example is the relay of BPM orbit data, which: (a) underwent serialisation through ROOT TInterlink objects; (b) sent via UDP to the OFSU; (c) and finally set to the appropriate OFSU FESA property for orbit data. The OFSU was responsible for logging the data collected by the OFC along with the settings of the BBFS itself. The OFSU was developed using the FESA real-time framework and the sheer number of tasks described above made the OFSU the largest FESA-based application at CERN until LHC Run 2 [8, 12].

In its initial implementation the BBFS comprised a series of nested feedback loops, each controlling a specific LHC parameter. First and foremost was the orbit feedback loop, which calculated the change in Closed Orbit Dipole (COD) deflections necessary to steer the beam towards the reference orbit. Second was the RF frequency feedback loop, which calculated the change in frequency required in the RF cavities to counteract any systematic momentum offset introduced by the CODs during orbit correction. The other three nested feedback loops controlled the tune, coupling and chromaticity simultaneously. It is important to note that in the latest implementation of the OFC, only the QFB remained. Due to the instabilities observed in the tune estimates, subsequently derived measurements, namely coupling and chromaticity, were not reliable enough to allow for feedback loops of their own. When adjustments on the coupling and chromaticity were required, operator-calculated corrections had to be sent manually to the skew quadrupoles and sextupoles [13].

The impact of a change of currents in the CODs on the beam orbit is estimated using a Response Matrix (RM). The main principle behind the orbit correction performed by the OFC is to: (a) measure the average beam positions using the BPMs; (b) calculate its difference with respect to the reference orbit; (c) use a Proportional-Integral (PI) controller as a feedback loop; (d) calculate the change in current needed in each COD to correct the beam position; (e) scale down all the currents by the same factor to accommodate the slowest acting magnet and finally; (f) send the current corrections to the CODs. A similar procedure is used by the QFB to correct the tunes in both planes for both beams using quadrupole correctors.

Figure 3 shows the flowchart of the OFC main program. `GLOBAL_RUN` was the variable which controlled the control loop. The `INIT` block is expanded in Fig. 4 where it can be seen that several initialisation files were required to set up the OFC. The configuration file set up the thread parameters such as their priority and affinity and also contained two important parameters related to testing the OFC; `SIM_MODE` and `BASE_PORT`. `SIM_MODE` were used extensively throughout all the OFC code to change its behaviour during testing. Functionalities such as sending the UDP packets to the Power Converters (PCs) and RF cavities were suppressed when `SIM_MODE` was set in order to confine the output of the OFC to the testing framework. `BASE_PORT` served as an offset port number for all the network sockets used by the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

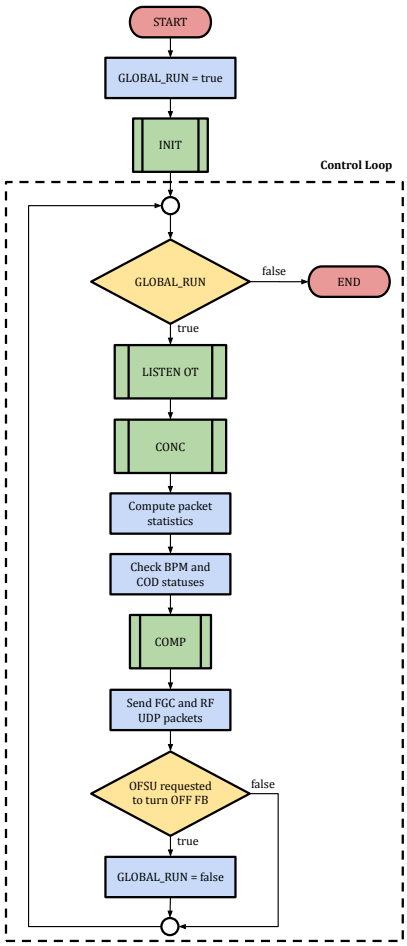


Figure 3: Flowchart of the main program of the OFC.

OFC. It was only changed during testing, when data was supplied by the testing framework. The configuration file contained other parameters, however, their use had become deprecated throughout the OFC lifetime.

The OFC also required a ROOT file upon initialisation which contained default optics stored in a ROOT object. This file existed since the first version of the OFC and due to its age it proved difficult to verify its contents. Its removal was also deemed dangerous due to its ubiquitous use throughout the initialisation code and therefore it was left up to the next major renovation to be removed. Several other text files were also required upon initialisation which provided: a) the hostnames of all the FECs that the OFC communicated with; b) the parameters of the magnets, e.g. maximum current rate; c) network routing information for all the BPMs, CODs, and quadrupoles required to create the network packets.

The OFSU was the interface that the operators used during real operation in order to monitor the status of the OFC, and changing parameters needed for operation. Two of the main problems with the OFSU were that it was bloated with unused functionality as well as complicated and less intuitive procedures which were required to change certain settings, e.g. fetching optics. The early design of the BBFS allowed access to more dedicated settings in the OFC, however, operator experience throughout the years has dictated which functionality to keep and improve, and which is not relevant for operations. On a final note, to avoid changing the major release of the BBFS during Run 2, any code renovations were delayed until LS2.

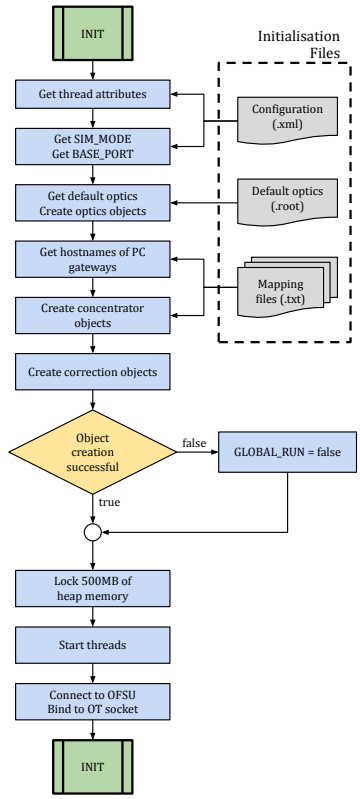


Figure 4: Flowchart of the `INIT` block in Fig. 3.

CODE RENOVATION

During LS2 it was decided to condense all the code in the BBFS, into one FESA-based application called BFCLHC. The hardware requirements were no longer an issue following a hardware upgrade of the OFC and OFSU machines. The hardware upgrade was from a 24-core, 32 GB RAM, 15 MB cache machine to a 64-core, 200 GB RAM and 22MB cache machine [14]. This upgrade meant that the conservative measures taken in the initial design of the OFC could be relaxed. During LHC operation, the OFC never crashed due to insufficient hardware requirements, therefore the hardware upgrade meant that it was safe to implement more advanced code and retain a deterministic execution.

At the same time, the FESA framework was undergoing development. With the introduction of new features, most of the basic operations done by the BBFS until Run 2 could be replaced by simple FESA commands. Some of the code used in the OFC and QFB was left intact: a) concentration of the COD, BPM and BBQ data; b) computation of the corrections required on the CODs and the tuning quadrupoles; c) Sending of the corrections to the respective gateways. The main program shown in Fig. 3 and the OFSU FESA class,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

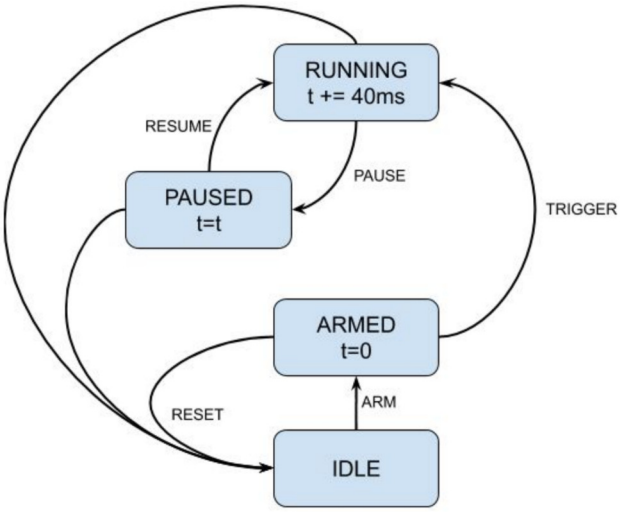


Figure 5: State machine of the FP as implemented in the BFCLHC.

contained the majority of the code which was replaced. The FESA framework was used to re-implement the logic of the main program.

To improve maintainability, the BFCLHC was designed to have a minimal amount of user code by relying more on the FESA framework features to implement the basic logic of the BFCLHC. For example, the data concentration is performed in a Real-Time Action (RTA), which updates the shared memory of the device with the latest BPM orbit data. A standard FESA interface property is then set up for providing the most recent BPM orbit measurements directly from shared memory. Following this design approach, trivial tasks (e.g. relaying data from real-time devices to users) were easier to implement and to debug. Considering that back-compatibility is ensured by FESA, future versions of BFCLHC will be easier to maintain [15]. It is important to note that the design of the BFCLHC Application Programming Interface (API) was based on the API of the OFSU, however, it was adapted significantly to better fit the needs of the operators.

Function Players (FPs) was a new feature that was added to the BFCLHC, at the request of the LHC operators. Using FPs for automated settings control was introduced in [16]. In the BFCLHC, the FPs are used to automate the change of: a) reference values, e.g. reference orbit; b) control loop gains; and c) optics models. Presently, the BFCLHC can accept a list of reference values along with a list of time offsets when the references should be applied. Linear interpolation of reference values was also implemented, e.g. varying the reference orbit linearly. Each FP is used depending on its current state. Figure 5 shows the general state machine of the FPs as implemented in the BFCLHC. The state is stored in the FESA shared memory, and can be changed by a FESA command. To change the state of the FP, the operators must either manually send events, or synchronise them with LHC timing. Some events are only available in specific states and a FESA exception is thrown when the user violates the state machine rules shown in Fig. 5; e.g. when the FP is in the `ARMED` state, only a `TRIGGER` or `RESET` event can change its state. A function can only be set when the respective FP is in `IDLE` state.

The optics calculation procedures are now delegated to the BFCLHC class. The approach to initialising and loading new optics to the feedbacks has therefore been changed. Any optics model must now be added to the BFCLHC memory via a FESA set command. Subsequently, a FP must be used to instruct the feedbacks when to change to new optics. The operators rely on their own sub-routines which can obtain a set of optic models from LHC Software Architecture (LSA) settings database [17], and forward it to the BFCLHC.

Following the Hardware Acceleration (HA) feasibility study in [18], we concluded that the use of GPUs did not improve computation times when calculating the Pseudo-Inverse (PInv) of the RMs when compared to a multi-core approach. In the new design, after an optics model is added, a new thread is spawned which calculates the Response Matrix (RM) as well as the corresponding PInv. In the event of a sensor or magnet malfunction, a thread per optic model can be spawned to re-calculate the adjusted models without blocking the control loop. The time to perform an optics recalculation took in the order of minutes to complete on the BBFS. The BFCLHC reduces this time to the order of seconds, which makes it possible to attempt in real-time in the LHC Run 3.

TESTING FRAMEWORK

The first formal testing of the OFC occurred at the start of the LHC Run 2 in 2014/15, during a code renovation that saw to the porting of the OFC and OFSU code from a 32-bit to a 64-bit architecture [19]. Considering the maintenance and correct porting of more than 90,000 lines of code, our colleagues from the operations group have built a separate testing framework during LS2 which mimics the input signals of the BBFS, by sending UDP packets disguised as coming from the BPM FECs.

The original design of the OFC made this operation complex due to its global control loop architecture. A series of code changes had to be made to the OFC and OFSU themselves simply to be able to test them. In particular, a global variable controlled key mechanisms within the OFC which determined whether the code was being used operationally or within a simulation. In addition, the output network ports were also tweaked in order to allow for a different base port to be used during testing. This was done as a protective measure against UDP packets being sent to the magnets during testing.

One of the main goals of the first testing framework was to verify that the renovated code respected all operational boundaries of the OFC. An identical hardware setup which was set up as a back-up to the operational hardware, had to be used to run the tests. The tests were written in a Java-embedded Domain Specific Language (eDSL) and the JUnit

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

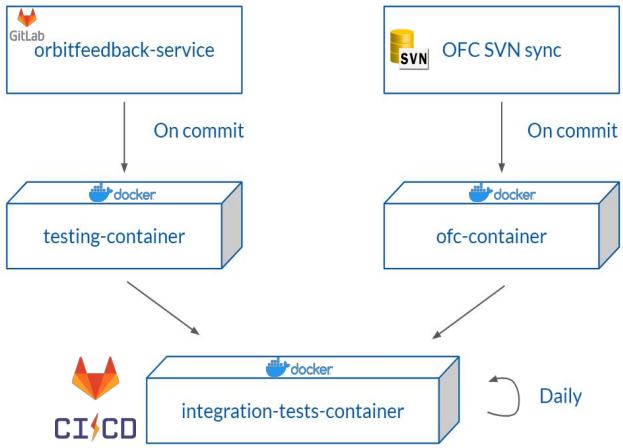


Figure 6: Testing framework schematic, showing the use of Docker containers in the GitLab Continuous Integration (CI) framework.

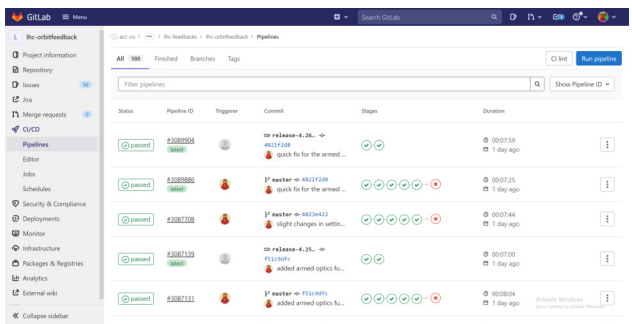


Figure 7: Screenshot of the BFCLHC testing framework test results using CI on GitLab.

framework was used to create the tests and generate reports. Due to the inherent complexity of this testing framework it was quickly realised that it was difficult to maintain and to use efficiently. Despite these limitations, this testing framework was successful in detecting bugs which aided the code renovation prior to the LHC Run 2.

The new BFCLHC testing framework is still undergoing development, and will continue at least until the start of Run 3. Figure 6 shows the basic structure of the latest design. It was decided to make the testing framework hardware agnostic with the use of Docker containers [20]. This new design allowed tests to be performed on the BFCLHC during active development through the use of Continuous Integration (CI) tools in GitLab [21]. As an example, algorithm 1 shows the pseudo-code for a test that checks the limit on the number of optics that can be loaded to the BFCLHC. Figure 7 is a screenshot from GitLab CI which shows the results of the tests that were performed automatically after successful compilation and Docker image creation.

CONCLUSION

The BBFS implemented the automatic feedbacks on the RF frequency, beam orbit and the tunes until LHC Run 2 in 2018. The renovation of the BBFS during LS2 saw the upgrade of the hardware, software and operational sides of the feedbacks. The experience gained when operating the feedbacks in the past, was taken into consideration during the renovation of the BBFS. Ultimately, the renovated BBFS is a FESA-based application called the BFCLHC, which includes all the required functionality of the BBFS, along with the addition of new features. The three main upgrades are: a) Introduction of FPs for the automatic changing of settings during LHC operation; b) More efficient optics computations and; c) a simpler and more user-friendly interface. A hardware agnostic testing framework was also developed by the LHC operations section to help with the development of the BFCLHC.

Algorithm 1 Test which asserts that only 30 optics can be loaded to BFCLHC.

```
optics_set                          ▷ Contains set of 30 optics
bfc                                 ▷ Running instance of BFCLHC
extra_optics                        ▷ 1 extra optics
res                                 ▷ Test result
for optics in optics_set do
    bfc.loadOptics(optics)
    if Allowable time expired then
        res ← Fail
        Exit
    else if not bfc.opticsLoadedSuccessfully() then
        res ← Fail
        Exit
    end if
end for
bfc.loadOptics(extra_optics)
if bfc.opticsLoadedSuccessfully() then
    res ← Fail
else
    res ← Success
end if
```

REFERENCES

[1] European Organization for Nuclear Research, O. Bruning, and European Council for Nuclear Research, *LHC Design Report, Volume I: The LHC Main Ring*, en. CERN, 2004. http://books.google.com/books/about/LHC_Design_Report_Volume_I.html?hl=&id=n-BmNQAACAAJ

[2] R. J. Steinhagen, "LHC beam stability and feedback Control-Orbit and energy," Ph.D. dissertation, RWTH Aachen U., Sep. 2007. http://cds.cern.ch/record/1054849

[3] L. K. Jensen *et al.*, "Software architecture for the LHC Beam-Based feedback system at CERN," CERN, San Francisco, USA, Tech. Rep. CERN-ACC-2013-0257, Nov. 2013. https://cds.cern.ch/record/1628552/files/CERN-ACC-2013-0257.pdf

[4] P. Forck, D. Liakin, and P. Kowina, "Beam position monitors," in *CERN Accelerator School: Beam Diagnostics*, D. Brandt, Ed., Dourdan, France: CERN, 2009, pp. 187–228. https://cds.cern.ch/record/1213277/files/p187.pdf

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV019

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[5] M. Gasior and R. Jones, "High sensitivity tune measurement by direct diode detection," in *Proceedings of 7th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC 2005)*, Lyon, France, 2005, p. 4. https://cds.cern.ch/record/895142/files/ab-2005-060.pdf

[6] S. Kostoglou, G. Arduini, L. Intelisano, Y. Papaphilippou, and G. Sterbini, "Impact of the 50 hz harmonics on the beam evolution of the large hadron collider," Feb. 2020. arXiv: 2003.00140 [physics.acc-ph]. http://arxiv.org/abs/2003.00140

[7] L. Grech *et al.*, "An alternative processing algorithm for the tune measurement system in the LHC," in *Proceedings of the 9th International Beam Instrumentation Conference (IBIC 2020)*, Virtual, 2020.

[8] M. Arruat *et al.*, "Front-end software architecture," in *ICALEPCS07*, vol. 7, Knoxville, Tennessee, USA, 2007, pp. 310–312. https://accelconf.web.cern.ch/accelconf/ica07/PAPERS/WOPA04.PDF

[9] R. Murillo-Garcia, Q. King, and M. M. De Abril, "Control of fast-pulsed power converters at CERN using a function generator controller," CERN, Tech. Rep. CERN-ACC-2015-0128, Oct. 2015. https://cds.cern.ch/record/2059133/files/CERN-ACC-2015-0128.pdf

[10] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," *Nucl. Instrum. Meth. A*, vol. 389, M. Werlen and D. Perret-Gallix, Eds., pp. 81–86, 1997. DOI: 10.1016/S0168-9002(97)00048-X.

[11] *Release 5.34/20 - 2014-08-13 | ROOT a data analysis framework*, https://root.cern.ch/content/release-53420, Accessed: 2019-7-31. https://root.cern.ch/content/release-53420

[12] J. Wenninger and R. Steinhagen, "LHC orbit feedback control requirements," CERN AB-OP, Tech. Rep., Mar. 2007.

[13] J. Wenninger, L. Grech, Ed., Personal communication, CERN, Jun. 2018.

[14] R. Jones, *BE-BI 2019: The year in review*, BI Day 2019, La Villa du Lac, Divonne-les-Bains, France, 2019. https://indico.cern.ch/event/857941/contributions/3612395/attachments/1960206/3257519/BI_Group_End_of_year_2019_-_compressed.pdf

[15] F. W. Hoguin, *FESA quality assurance*, 1st Developers@CERN Forum, CERN, Geneva, Switzerland, Sep. 2015. https://cds.cern.ch/record/2056256

[16] D. Alves, K. Fuchsberger, S. Jackson, and J. Wenninger, "Test-driven software upgrade of the LHC beam-based feedback systems," in *2016 IEEE-NPSS Real Time Conference (RT)*, Padova, Italy: IEEE, Jun. 2016. https://indico.cern.ch/event/390748/contributions/1825184/attachments/1283146/1927869/CRX_PosterSession2_64.pdf

[17] C. Roderick and R. Billen, "The LSA database to drive the accelerator settings," Tech. Rep. CERN-ATS-2009-100, Nov. 2009. https://cds.cern.ch/record/1215575?ln=en

[18] L. Grech, D. Alves, S. Jackson, G. Valentino, and J. Wenninger, "Feasibility of hardware acceleration in the LHC orbit feedback controller," en, in *17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, 05-11 October 2019*, JACOW Publishing, Geneva, Switzerland, Aug. 2020, pp. 584–588. https://accelconf.web.cern.ch/icalepcs2019/doi/JACoW-ICALEPCS2019-MOPHA151.html

[19] S. Jackson, D. Alves, L. Di Giulio, K. Fuchsberger, B. Kolad, and J. Pedersen, "Testing framework for the LHC beam-based feedback system," in *Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems*, L. Corvetti, K. Riches, and V. Schaa, Eds., Melbourne, Australia, Jan. 2016, pp. 140–144. https://accelconf.web.cern.ch/ICALEPCS2015/papers/mopgf024.pdf

[20] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, Mar. 2014. https://dl.acm.org/doi/10.5555/2600239.2600241

[21] *GitLab CI/CD*, https://docs.gitlab.com/ee/ci/, Accessed: 2021-8-31. https://docs.gitlab.com/ee/ci/

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV020

LEARNING TO LASE: MACHINE LEARNING PREDICTION OF FEL BEAM PROPERTIES

A.E. Pollard*, D.J. Dunning, M. Maheshwari
ASTeC, Cockcroft Institute, STFC Daresbury Laboratory, UK

Abstract

Accurate prediction of longitudinal phase space and other properties of the electron beam are computationally expensive. In addition, some diagnostics are destructive in nature and/or cannot be readily accessed. Machine learning based virtual diagnostics can allow for the real-time generation of longitudinal phase space and other graphs, allowing for rapid parameter searches, and enabling operators to predict otherwise unavailable beam properties. We present a machine learning model for predicting a range of diagnostic screens along the accelerator beamline of a free-electron laser facility, conditional on linac and other parameters. Our model is a combination of a conditional variational autoencoder and a generative adversarial network,which generates high fidelity images that accurately match simulation data. Work to date is based on start-to-end simulation data, as a prototype for experimental applications.

INTRODUCTION

Free-electron lasers (FELs) are sources of ultra-short and ultra-bright pulses of light, which are used to reveal the dynamics of important processes across various scientific disciplines [1]. For effective and efficient facility operation, it is necessary to rapidly reconfigure and optimise different setups to meet user needs. Machine diagnostics are essential for this task, and while some measurements can be taken non-invasively and on every shot, others are destructive to the beam, relatively slow, or otherwise constrained. For example, the longitudinal phase space (LPS) of the beam is critically important for FEL performance but this is an invasive measurement that can only be made at locations with dedicated hardware. Simulations are commonly used to supplement the experimental data; however, they can be computationally expensive and often require significant iteration to improve their match to experimental conditions. Machine learning offers the potential to leverage the large volumes of accessible diagnostic and simulation data to build surrogate models that enable accurate modelling with real-time execution, thereby delivering virtual diagnostics and rapid optimisation of beam conditions on demand.

Artificial Neural Networks have shown great utility in the generation of complex data. From synthesised speech to faces of non-existent people, several architectures have been developed to effectively generate artificial data that is indistinguishable from the training data. Autoencoders, autoregressive models, and generative adversarial networks comprise the three main classes of these generative networks and the focus of modern research.

* amelia.pollard@stfc.ac.uk

Autoencoders are an excellent choice for generating complex and variable data in an unsupervised manner. Their ability to condense inputs into a significantly lower dimensional representation before reconstructing the input from this representation provides a powerful method of learning the key components of target images, such as input parameters. Variational autoencoders [2] build on this to codify the latent space representation as a series of probability distributions which can be further extended by making those distributions conditional on input parameters. This Conditional Variational Autoencoder(CVAE) model enables effective learning of the parameterised generation of images.

Generative Adversarial Networks [3](GAN) represent the current state-of-the-art approach to image generation and function on a principle of adversarial learning. Adversarial learning functions by one network attempting to generate realistic data, while another network attempts to discern real data from generated data. This technique enables image generation with fidelity orders of magnitude higher than that of generative networks alone, but it is prone to stability issues and is highly sensitive to hyperparameter choices.

Our work combines a CVAE and a GAN into the CVAE-GAN architecture [4] and uses a combination of these powerful techniques to produce high-fidelity longitudinal phase space graphs for arbitrary parameter configurations as a virtual diagnostic for end users. We also demonstrate an ability to search the space of LPS graphs for particular graphs as drawn by an end user, allowing for highly customisable longitudinal beam profiles.

RELATED WORK

The first steps to incorporating image recognition into particle accelerator control using convolutional neural networks were taken in Edelen et al. [5] and developed further in Scheinker et al. [6]. Conversely, machine learning *prediction* of longitudinal phase space from machine parameters has been explored, with good results, as in Emma et al. (2018 and 2019) [7, 8]. While these simple networks seem to lead to artifacts and sub-optimal reconstruction of fine structure details, more complex networks featuring convolutional and upsampling layers do appear to provide some improvement [9]. The use of additional information besides machine settings, in the form of non-invasive shot-to-shot spectral measurements, has also been shown to improve the accuracy of LPS predictions [10]. Additionally, recent work based solely on experimental data has demonstrated LPS prediction at significantly higher resolution to previous studies [11].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV020

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

METHODOLOGY

We used a dataset of longitudinal phase space images and associated parameters, as described in [12] and briefly outlined here. This was generated from the results of previous work, using a Multi-Objective Genetic Algorithm [13] to find optimal beam characteristics on the XARA lattice [14]. The XARA lattice is a possible upgrade to the existing CLARA facility [15], using high-gradient X-band RF technology. A diagram of the XARA lattice can be found in Fig. 1. 10,000 start-to-end simulations of the accelerator were generated using the simulation code ASTRA [16] (up to the linac 1 exit) and Elegant [17] for the remainder. Using the 6D bunch distributions produced, LPS images were created by calculating the maximum and minimum values of the z-positions and beam energies for each individual distribution and binning the particles in a 2D histogram defined by this region of interest (ROI) to create 100×100 pixel images. Though not used in this study, another set of images was generated for which the maxima and minima were taken over all distributions, for a fixed 'screen size' (referred to as non-ROI or fixed extent images).

Each example in the dataset consists of an 100×100 pixel LPS image at a location immediately upstream of the undulator line, 4 real values describing the bounds of the ROI in the fixed extent, and 17 accelerator parameters describing the phase and amplitude of the linacs, the laser heater factor, the dechirper factor, and the bunch compression.In this work, all fixed extent images shown were generated by placing the ROI image in accordance with the bounds generated by the network.

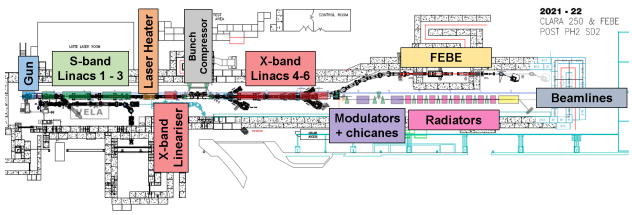


Figure 1: Schematic layout of CLARA with the potential XARA upgrade overlaid. The total length is approximately 90 m. The LPS images used in this study were generated at a location following the X-band linacs and before the modulator undulators.

Following the approach taken in previous studies [7], we began by developing a fully connected parameter to image generative model as part of Maheshwari et al. [12]. A parameter to CNN generative model was also tried, which performed poorly, generating approximations which were of the correct scale but lacking a well-defined structure. This work led directly to the development of this more complex architecture.

We then developed a conditional variational autoencoder model and experimented with CNN and DNN based generators to produce high quality images. However, as seen in previous works, the generator showed a tendency to produce a great many artifacts and fine details were lost. We therefore created a deep neural network which utilised a convolutional conditional variational autoencoder, with the addition of a discriminator, resulting in a model known as a CVAE-GAN (as shown in Fig. 2) [4]. A discriminator endeavours to identify real and generated images. When this is affixed to the generative model (the CVAE), the weights of the discriminator are frozen and the generated images are intentionally mislabelled as being real images. The resulting error is then backpropagated through the generative network to train the generator to produce images which would "fool" the discriminator. This error is combined with an absolute difference error metric between the input and output images, as well as the Kullback-Liebler divergence constraint on the latent space of the CVAE.

Hyperparameters were found by Bayesian optimisation to minimise absolute difference over a validation set composed of 10% of the total dataset. The network was trained utilising 90% of the dataset, with 10% held back for testing. The Adam optimiser was used with a learning rate of 1.0×10^{-3}, with the error function defined in Eq. (1). Note that D_{kl} represents the Kullback-Liebler divergence [18].

$$\begin{aligned} E(x) = D_{kl}(N(0,1)||P(x|\mu,\sigma,y)) \\ +abs(x-\bar{x}) \\ -(x_d log(\hat{x}_d) + (1-x_d) log(1-\hat{x}_d)) \end{aligned} \tag{1}$$

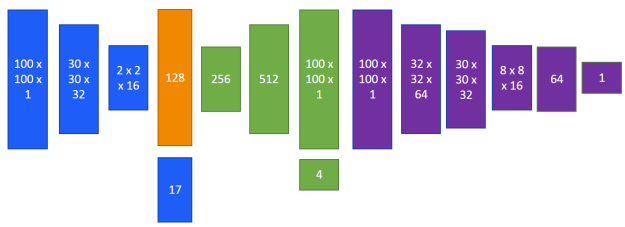


Figure 2: The model of the CVAE-GAN, with the encoder highlighted in blue, the latent space layer highlighted in orange, the decoder highlighted in green, and the discriminator highlighted in purple. Note the additional 1×4 output layer which generates the position of the region of interest in the fixed extent image.

RESULTS

We found that the CVAE-GAN model allowed for the real-time generation of longitudinal phase space graphs with high verisimilitude to simulation. This therefore achieves the stated goal of providing a virtual diagnostic of longitudinal phase space that would allow operators to carefully choose parameters to achieve specific bunch energy profiles.

Most notably, our technique delivers high quality representation of fine-grain details as can be seen in Fig. 3. This is a consequence of the application of the discriminator, which constrains the generator to produce more realistic images where more traditional error functions, such as absolute difference, generally result in blurring of fine details.

Given that the forward pass of neural networks can be run in linear time, this allows for the rapid sampling of LPS graphs from the space. In combination with Bayesian

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV020

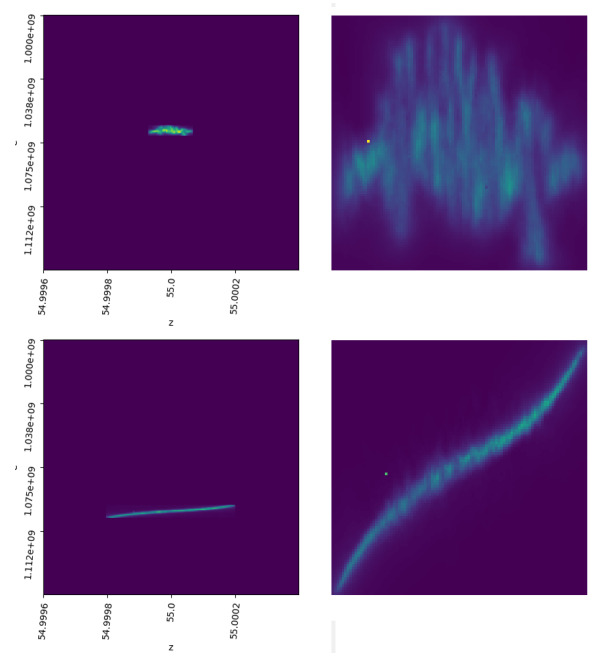


Figure 3: A pair of examples from the LPS GUI showing both the fixed extent and region of interest. Note the fine-grain details of the region of interest image, which are generally lost in simpler models.

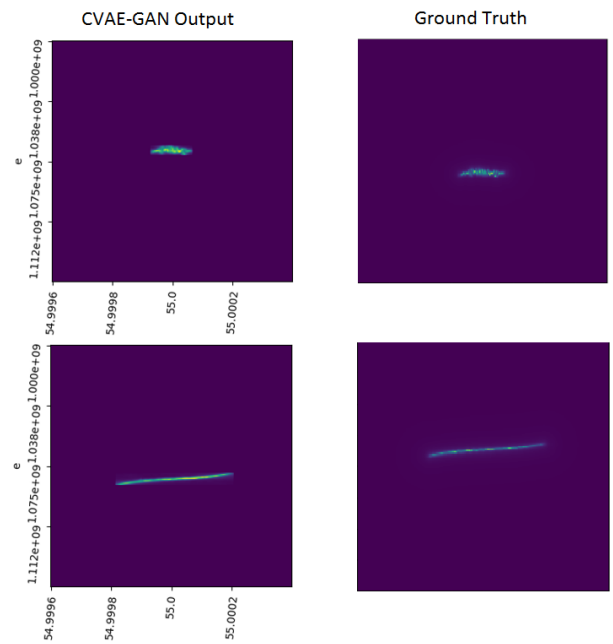


Figure 4: Fixed extent images and ground truth from Elegant simulations with the same parameters. Variations in the position of ROI are due to scale differences in the output. Significantly, the finer detail structure of the LPS graph is strongly matched to the output of Elegant.

Optimisation, we can therefore perform a guided search for image similarity with an LPS graph drawn by an operator, as shown in the example in Fig. 5, to yield the required machine settings. This feature is one part of a graphical user interface (GUI) for the model, shown in Fig. 6.

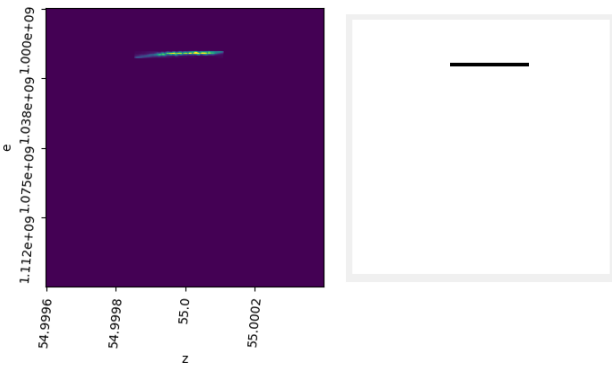


Figure 5: An example of the drawing interface with target drawn by operator (right) and the resulting LPS graph (left) which most closely matched the operator's drawing. On a mid-range laptop, this search took approximately 20 seconds.

CONCLUSION

Utilising the CVAE-GAN architecture allows for high fidelity linear time virtual diagnostics of the longitudinal phase space for arbitrary accelerator parameters. However, expanding of the dataset to encompass a wider range of parameters significantly increases the quantity of data required to provide an accurate surrogate model. In continuing this research to expand this work to multiple screen locations along the beamline and to a wider parameter space, we have found that this increase in the generative range of the model has led to common Generative Adversarial Network pitfalls such as mode collapse and instability. We therefore conclude that this technique, while extremely powerful, is also quite difficult to implement reliably and care should be taken to avoid such issues.

Future work will include the simultaneous generation of LPS graphs for multiple points along the beamline, as well as comparisons to real-world data using a transverse deflecting cavity and potentially applying transfer learning using a small quantity of TDC data to augment a larger quantity of simulation data.

REFERENCES

[1] E. A. Seddon *et al.*, "Short-wavelength free-electron laser sources and science: A review," *Reports on Progress in Physics*, vol. 80, no. 11, p. 115 901, 2017, doi:10.1088/1361-6633/aa7cca

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[3] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[4] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "CVAE-GAN: Fine-grained image generation through asymmetric training," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2745–2754.

[5] A. L. Edelen, S. Biedron, S. V. Milton, and J. P. Edelen, "First steps toward incorporating image based diagnostics into particle accelerator control systems using convolutional neural networks," in *Proc. North American Particle Accelerator*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV020

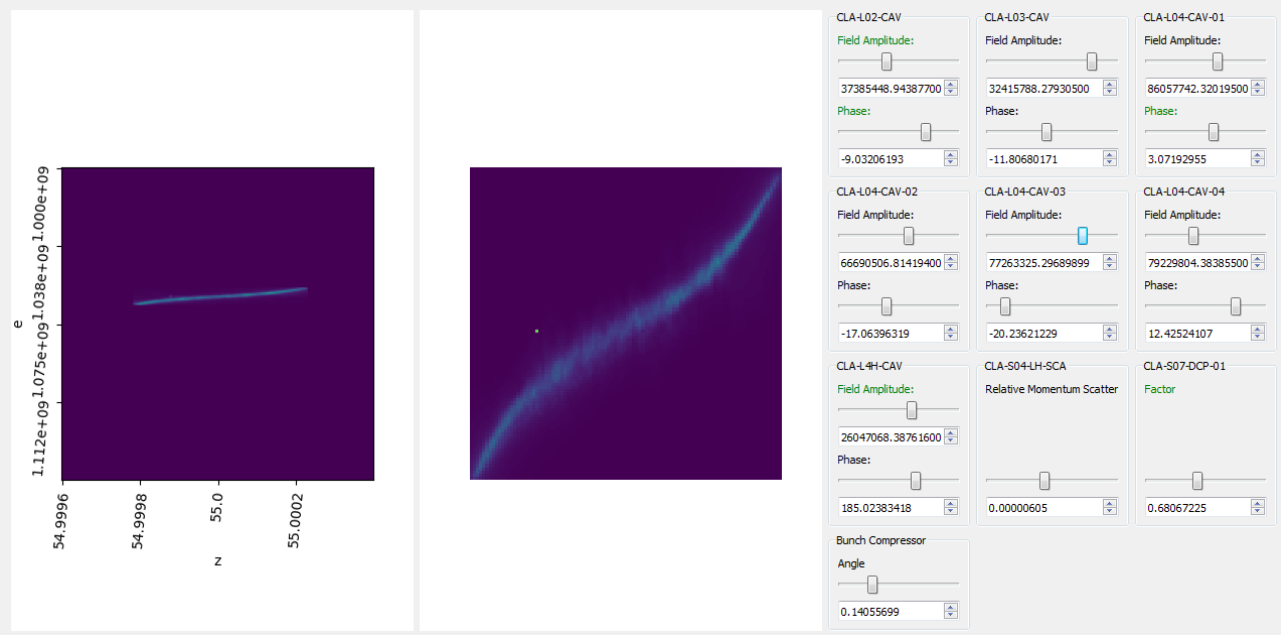


Figure 6: LPS CVAE-GAN GUI showing an example configuration, the fixed extent of the longitudinal phase space graph, and the region of interest which shows finer detail of the LPS distribution. Note the colouring of the labels above the sliders, which indicates the sensitivity of the LPS graph to that parameter in this region, with green indicating stability and red indicating significant impact from changes to this parameter.

Conf. (NAPAC'16), Chicago, IL, USA, 2016, pp. 390–393, doi:10.18429/JACoW-NAPAC2016-TUPOA51

[6] A. Scheinker, A. Edelen, D. Bohler, C. Emma, and A. Lutman, "Demonstration of model-independent control of the longitudinal phase space of electron beams in the linac-coherent light source with femtosecond resolution," *Phys. Rev. Lett.*, vol. 121, p. 044 801, 4 2018, doi:10.1103/PhysRevLett.121.044801

[7] C. Emma, A. Edelen, M. Hogan, B. O'Shea, G. White, and V. Yakimenko, "Machine learning-based longitudinal phase space prediction of particle accelerators," *Physical Review Accelerators and Beams*, vol. 21, no. 11, p. 112 802, 2018.

[8] C. Emma *et al.*, "Machine learning-based longitudinal phase space prediction of two-bunch operation at FACET-II," SLAC National Accelerator Lab., Menlo Park, CA, USA, Tech. Rep., 2019.

[9] A. Edelen, N. Neveu, C. Emma, D. Ratner, and C. Mayes, "Machine learning models for optimization and control of x-ray free electron lasers," in *Proc. NeurIPS Machine Learning for the Physical Sciences Workshop (NeurIPS2019)*, Vancouver, Canada, 2019.

[10] A. Hanuka *et al.*, "Accurate and confident prediction of electron beam longitudinal properties using spectral virtual diagnostics," *Scientific Reports*, vol. 11, no. 1, pp. 1–10, 2021.

[11] J. Zhu, Y. Chen, F. Brinker, W. Decking, S. Tomin, and H. Schlarb, "High-fidelity prediction of megapixel longitudinal phase-space images of electron beams using encoder-decoder neural networks," *Phys. Rev. Applied*, vol. 16, p. 024 005, 2 2021, doi:10.1103/PhysRevApplied.16.024005

[12] M. Maheshwari, D. Dunning, J. Jones, M. King, H. Kockelbergh, and A. Pollard, "Prediction and Clustering of Longitudinal Phase Space Images and Machine Parameters Using Neural Networks and K-Means Algorithm," in *Proc. IPAC'21*, Campinas, SP, Brazil, 2021, paper WEPAB318, pp. 3417–3420, doi:10.18429/JACoW-IPAC2021-WEPAB318

[13] D. Dunning, H. C. Cortés, J. Jones, and N. Thompson, "Multi-Objective FEL Design Optimisation Using Genetic Algorithms," in *Proc. FEL'19*, Hamburg, Germany, 2019, pp. 711–714, doi:10.18429/JACoW-FEL2019-THP065

[14] D. Dunning, L. Cowie, and J. Jones, "XARA: X-Band Accelerator for Research and Applications," in *Proc. FEL'19*, Hamburg, Germany, 2019, pp. 715–718, doi:10.18429/JACoW-FEL2019-THP066

[15] D. Angal-Kalinin *et al.*, "Design, specifications, and first beam measurements of the compact linear accelerator for research and applications front end," *Phys. Rev. Accel. Beams*, vol. 23, p. 044 801, 4 2020, doi:10.1103/PhysRevAccelBeams.23.044801

[16] K. Floettmann, "Astra - a space charge tracking algorithm," Tech. Rep., http://www.desy.de/~mpyflo

[17] M. Borland, "Elegant: A flexible SDDS-compliant code for accelerator simulation," Argonne National Lab., IL (US), Tech. Rep., 2000.

[18] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV021

MACHINE LEARNING FOR RF BREAKDOWN DETECTION AT CLARA

A. E. Pollard*, A. J. Gilfellon[1], D. J. Dunning
ASTeC, Cockcroft Institute, STFC Daresbury Laboratory, Warrington, UK
[1]also at The University of Manchester, Manchester, UK

Abstract

Maximising the accelerating gradient of RF structures is fundamental to improving accelerator facility performance and cost-effectiveness. Structures must be subjected to a conditioning process before operational use, in which the gradient is gradually increased up to the operating value. A limiting effect during this process is breakdown or vacuum arcing, which can cause damage that limits the ultimate operating gradient. Techniques to efficiently condition the cavities while minimising the number of breakdowns are therefore important. In this paper, machine learning techniques are applied to detect breakdown events in RF pulse traces by approaching the problem as anomaly detection, using a variational autoencoder. This process detects deviations from normal operation and classifies them with near perfect accuracy. Offline data from various sources has been used to develop the techniques, which we aim to test at the CLARA facility at Daresbury Laboratory. These techniques could then be applied generally.

INTRODUCTION

There are two main aims with this project. Firstly, we aim to assemble a machine learning (ML) based system that could be used to replace the current mask method of radio frequency (RF) breakdown (BD) detection which is standard in the automated code used in the RF conditioning of accelerating cavities. Secondly, we aim to ensure that the mid-process features of the same mechanism could be used as inputs for an ML algorithm designed to predict whether or not the next RF pulse would lead to a BD.

To this end, we constructed a β convolutional variational autoencoder (βCVAE)[1] with RF conditioning data as inputs. After being trained as an anomaly detector this acted as a live BD detector, in conjunction with a dense neural network (NN), which would act with the capacity to replace the current non-ML based BD detection system. In addition to this, the βCVAE's latent space could act as a viable input for a long short-term memory (LSTM) recurrent neural network (RNN) that could be used to predict BDs, based on the methodology set out by Kates-Harbeck et al.[2] who had success in predicting disruptive instabilities in controlled fusion plasmas.

For this investigation, we used data from the CLARA accelerator (Compact Linear Accelerator for Research and Applications) based at Daresbury Laboratory. CLARA is a dedicated accelerator test facility with the capacity to deliver high quality electron beams for industry and research. In addition to the CLARA data, a larger dataset was provided by the CLIC team at CERN covering a cavity test which took place in CERN's XBOX-2 test stand. The structure tested in this dataset was a T24 high-gradient prototype X-band cavity produced at the Paul Scherrer Institute; further details of this design have been reported previously [3, 4]. The CLARA data was collected as part of the routine RF breakdown detection system.

* amelia.pollard@stfc.ac.uk

RELATED WORK

Solopova et al.'s [5] application of a decision tree model to assign both a fault type and cavity-specific location to a collected breakdown signal at CEBAF represents the first foray into using machine learning to classify RF cavity faults. This work was then continued in Tennant et al.[6] where the authors applied a random forest model to the classification of faults and cavity identity for a larger dataset of breakdown events.

Obermair et al. [7] took the first step towards machine learning based detection and prediction of breakdowns. The authors separately applied deep learning on two available data types (event and trend data) to predict breakdowns. In so doing, they were able to predict breakdowns 20ms in advance with good accuracy. In addition, they utilised explainable AI on these models to elucidate the physics of a breakdown. This pointed them towards an increased pressure in the vacuum system before a breakdown, which they indicated as an option for an improved interlocking system. Their analysis of event data alone also reveals the possibility to predict breakdowns with good accuracy, if there has already been a breakdown in the previous minute, i.e. prediction of follow-up breakdowns.

Previous work within our organisation also informed the present studies. Another dataset from XBOX cavity testing was analysed for missed breakdowns using principal component analysis and neural networks [8]. Very high classification accuracy was reported, but there was suspected duplication of traces in the dataset.

METHODOLOGY

CLARA

Here we use data gathered during the RF conditioning of CLARA's 10 Hz photoinjector (Gun-10), which includes both the RF pulse traces themselves and other non-RF, such as the temperature and pressure inside Gun-10. The RF trace data was gathered before ML was taken into consideration and was therefore not ideal for our purposes, but it was deemed to be sufficient for progress to be made. The trace data was only recorded when the RF breakdown detector was activated and a BD identified, then the conditioning script

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

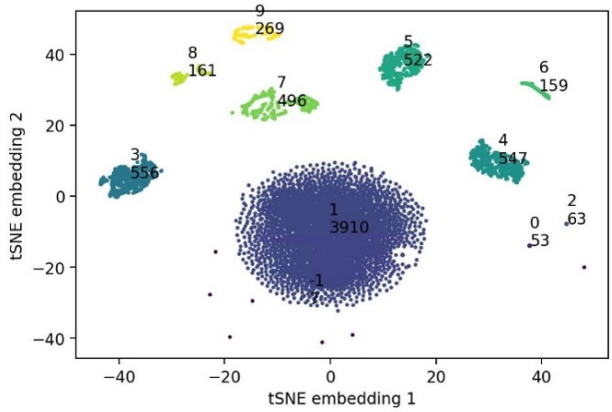


Figure 1: A plot of the results of applying the t-SNE/DBSCAN method to label the CLARA Gun-10 data set. Each cluster is given an index with its population displayed below, i.e. the cluster in the top right corner of the plot has an index of 6 and a population of 159. As examples of the principal trace types, the large central cluster indexed as 1 represents noise traces, cluster 4 contains only healthy traces, 8 breakdown traces, and 7 is composite (healthy/BD).

would record the pulse associated with the BD, as well as the two previous and two subsequent pulses. Altogether, there were 40 traces recorded per breakdown event (8 traces for each of the 5 pulses). Specifically the traces were: klystron (forward, reverse power, and phase), and cavity,(forward, reverse power, and phase).

In order to provide the ground truth and label each recorded trace as either, noise, healthy, BD or anomaly, the traces were first grouped together by using sklearn's t-SNE[9] (t-distributed Stochastic Neighbour Embedding) and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) functions. The traces from each delivered cluster were then over-plotted and inspected by eye with any pure groups receiving the appropriate label and composite clusters undergoing further t-SNE/DBSCAN analysis until only pure groups remained. Figure 1 shows an example of the clustering that was returned by the t-SNE/DBSCAN method for this data set.

The next step was to construct the βCVAE at the core of the BD detector and at the beginning of the planned BD predictor. After much experimentation with spectrograms, phase traces, and temperature and pressure data, it was found that the most effective input for the βCVAE was a 2D array comprised of the four normalised power traces, with dimensions of 4×1017. The most optimal structure of the found for βCVAE is displayed and described in Fig. 2.

The βCVAE was trained on 4659 healthy traces in order to create the anomaly detector, which was then validated using another 1164 healthy traces. For both training and validation the Adam optimiser and categorical cross entropy loss function were used. Testing the βCVAE involved exposing the algorithm to 706 healthy and 706 BD traces (1412 traces in total) and subtracting the reconstructed traces from the original traces to produce 1D reconstruction error traces. These were then used as an input for a simple dense neural network classifier with one ReLU activated hidden layer with the same dimensions as the input layer and a binary (healthy or BD) softmax activated output layer. Again the Adam optimiser and categorical cross entropy loss function were used.

A confusion matrix was then constructed by comparing the class assigned by the model to the ground truth in order to check for the accuracy and recall of the BD row for the system, as in Table 2. For this dataset, the key statistic was the recall of the BD row, since the accelerator not reacting to a false negative could be damaging to the accelerating structure, whereas reacting to a false positive merely results in a slight reduction in the time efficiency of the accelerator. An accuracy of 96.9% and a BD row recall of 98.0% was achieved using the methods outlined above. We also noted that approximately half of the false negative traces were in fact healthy or anomalous after manual inspection. This is not surprising since the labelling process relied on unsupervised ML processes and, had time allowed, all traces would have been labelled individually by an RF expert.

Future work will include the integration of the ML BD detector into the next version upgrade of the RF conditioning code and the construction of the LSTM RNN for the predictive system. However, before this can be effective, more appropriate data may need to be gathered from CLARA, particularly more traces before a BD event and at a higher RF repetition rate than 10 Hz. Since CLARA RF power is pulsed if we were to follow the methodology set out in Kates-Harbeck et al.[2] we can think of the noise between pulses as missing data . In order to quantify the proportion of data that is effectively missing, or pseudo-missing, we can use the following relation,

$$R_{\mathrm{DATA}} = 1 - D, \tag{1}$$

where R_{DATA} is the proportion of the data that is pseudo-missing and D, the compliment of R_{DATA}, is the dimensionless duty cycle of the RF system, defined as,

$$D = \mathrm{PRR} \times \tau_{\mathrm{pulse}}, \tag{2}$$

where PRR is the pulse repetition rate in Hz and τ_{pulse} is the RF pulse length in seconds. For CLARA's Gun-10 we have a PRR of 10 Hz and an operational pulse length of 2.5 µs, which gives a duty cycle of 2.5×10^{-5}, or 0.0025% and consequently a proportion of pseudo-missing data of 0.999975, or 99.997%. It seems reasonable to assume that for any given system there may exist a "duty cycle threshold" below which BD prediction using ML with RF trace data is not practically possible and which may only be determined experimentally.

CERN

The CERN XBOX data consisted of two primary data types: event and trend. The trend data contained environmental data concerning the test cavity (i.e. temperature,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV021

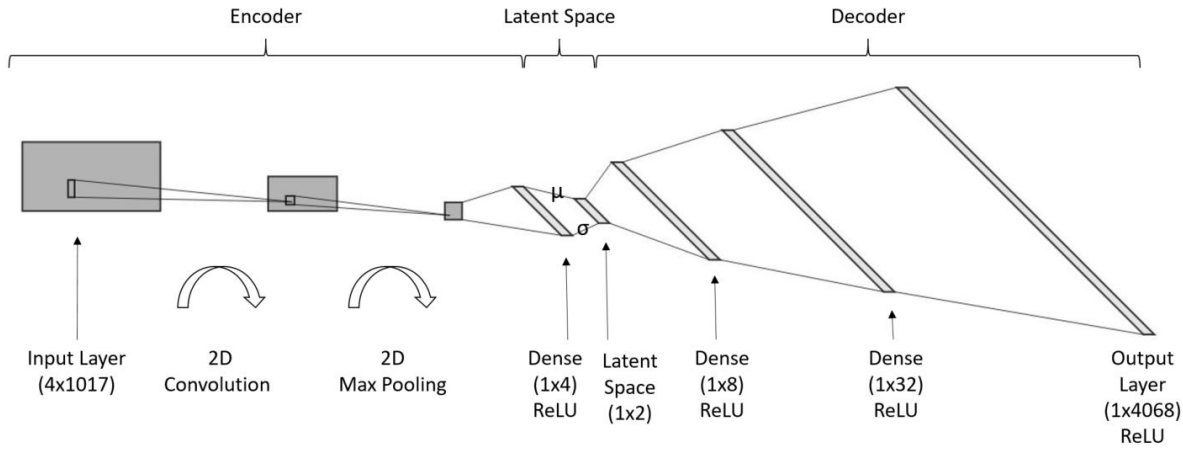


Figure 2: The final structure of the βCVAE for the CLARA dataset study. The encoding half of the βCVAE was constructed by using 2D arrays consisting of the four power traces (4×1017) as inputs to the first 2D ReLU activated convolutional layer with subsequent 2D max pooling layer, the arrays were then flattened and inputted into a 1×4 ReLU activated dense layer before being passed into the 1×2 latent space. The decoder consisted of two dense ReLU activated layers of dimensions 1×8 and 1×32, before a sigmoid activated output layer of 1×4068 ($= 4 \times 1017$). The outputted 1D array was then reshaped into a 4×1017 2D array in order to produce the reconstructed traces with the original input layer dimensions.

vacuum pressure) while the event data contained the signal traces from a number of components of the RF system.

To keep consistency with the CLARA breakdown detector, the trend data was discarded and only the RF traces were used. These traces consisted of 16 channels, of which we excluded 7, as follows. Two of the channels ('DC UP' and 'DC DOWN') corresponded to the Faraday cups upstream and downstream of the RF cavity, and these were used for automated labelling of the samples. A reading of less than -0.05 from either cup would mark that trace as a breakdown. The remaining 5 excluded channels were removed as they were either repeated signals (i.e. the 'PSR log' channel repeated 'PSR Amplitude' channel, but with log scaling) or essentially noise (such as the Beam Loss Monitor signal, which was indistinguishable from noise).

Further filtering was applied to ensure the quality of the data, and traces were removed wherein the mean and variance of the amplitude traces indicated that the RF cavity was not active ($\bar{x} \times \sigma^2(x) < 1e{-}4$), and the signals were therefore considered noise. This filtering and classification resulted in 254,656 samples, of which 5,930 samples contained a breakdown.

We then developed a beta variational autoencoder model, as shown in Fig. 3, to reconstruct only healthy signals. The network was trained using 90% of the healthy signals. The Adam optimiser was used with a learning rate of $1e-3$, with the error function defined in equation (3). The network was trained to convergence, which took 27 epochs. For β a value of 5 was chosen by grid search.

$$E(x) = \beta D_{kl}(N(0,1)||P(x|\mu,\sigma)) + abs(x - \bar{x}) \quad (3)$$

Note that D_{kl} represents the Kullback-Liebler divergence[10].

Once sufficiently trained to reconstruct healthy signals, we utilised this *overfitting* to detect breakdown events as anomalies. That is to say, when the network fails to reconstruct the signal well, we can be reasonably assured that this represents a deviation from normal operation and thus a breakdown event.

In order to classify the breakdown events, we began by passing all breakdown events and an equal number of non-breakdown events through the autoencoder, taking the reconstruction error for each channel and compiling those values into a vector. Applying a K-nearest neighbour algorithm to the per-channel reconstruction error vector resulted in mediocre performance, as shown in Table 1. As such, we elected to implement a simple multi-layer neural network to perform the classification, while also concatenating the latent space representation of the example to the per-channel reconstruction errors to form the input vector. This new neural network was then trained to convergence in 20 epochs using the Adam optimiser, with a learning rate of $1e-2$ and a binary cross-entropy loss function. Results from this network are shown in Table 3.

Table 1: kNN Results on CERN XBOX Data

	Positive	Negative
True	89.6%	98.8%
False	10.4%	1.2%

Early work into prediction of breakdowns was undertaken by replacing the classification network with an LSTM which was trained on a label of time-until-breakdown. Unfortunately, this achieved little success with our most effective attempt achieving 73% accuracy at predicting breakdowns within 30 second windows. We hypothesise this is due to the sparse nature of the sampling of shots, and future work

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

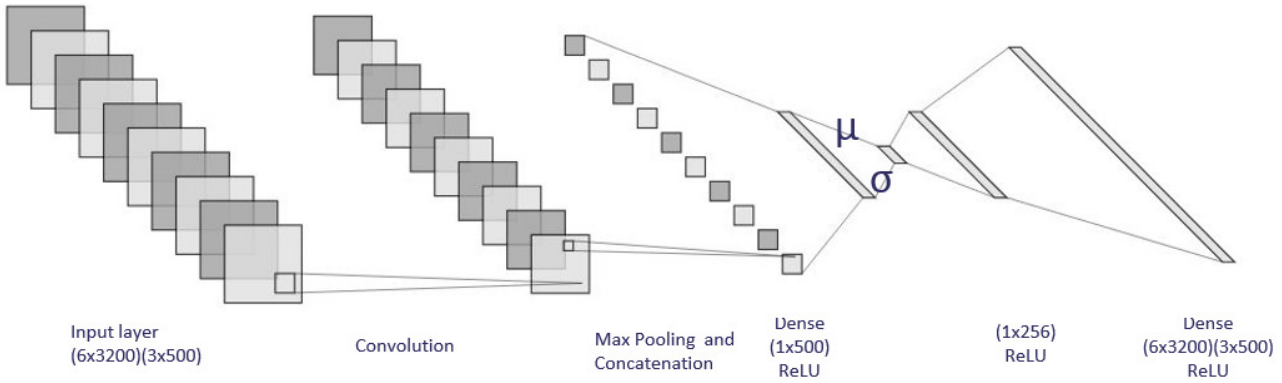


Figure 3: For the CERN XBOX data, three of the input channels were of a lower dimension, and each channel was processed by a CNN with filters of size 1×5, with the resulting feature vectors flattened and concatenated before being passed to a dense ReLu layer and encoded into the latent space. The decoder is constructed of a single dense ReLu layer with each channel consisting of a linear dense layer.

will involve collecting a higher frequency sampling of shots such that this prediction might be enabled.

RESULTS

As can clearly be seen, both networks produced strong results very high recall and accuracy. In particular, the high recall value is significant for such an unbalanced dataset. If of a system of this type were to be deployed to an edge ML system on an accelerator, a low false positive rate would be extremely important for operator trust of the system.

Table 2: Results on CLARA Data

	Positive	Negative
True	95.8%	98.0%
False	4.2%	2.0%

Table 3: Results on CERN XBOX Data

	Positive	Negative
True	97.9%	99.6%
False	2.1%	0.4%

It is of note that approximately half of the false positives in the CERN XBOX data are in fact true positives that were mislabelled by the automated labelling, as verified by manual inspection.

CONCLUSION

We find that the application of a variational autoencoder as an anomaly detector is extremely effective as a breakdown detector for RF cavities. A significant benefit of this approach is that it requires only healthy signals to train a strong detector, in contrast to supervised approaches which require careful balancing of positive and negative signals. Obermair et al.[7] showed that environmental trend data was sufficient to predict breakdowns with good accuracy up to 20ms in advance. They also found that shot traces are sufficient to predict breakdowns with good accuracy, if and only if there has already been a breakdown in the last minute. In agreement with their results, we find that the shot traces from the cavity alone are not sufficient at this sampling frequency to predict breakdowns ex nihilo. We plan to collect additional data from CLARA with complete shot capture. With this data, we hope to demonstrate breakdown prediction using only per-shot phase and amplitude traces and the anomaly detection method presented herein.

ACKNOWLEDGEMENTS

The authors would like to thank the CLIC team and the Paul Scherrer Institute for the provision of their high-gradient structure test data, in addition to Christoph Obermair and Lee Millar from CERN for their advice on interpreting it. We would also like to thank Hannah Kockelbergh (The University of Liverpool) for the studies that led on to this work, as well as STFC's Scientific Machine Learning (SciML) Group, particularly Keith Butler, for supervising A. J. Gilfellon's work on this project.

REFERENCES

[1] I. Higgins *et al.*, "Beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[2] J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," *Nature*, vol. 568, no. 7753, pp. 526–531, Apr. 2019. DOI: 10.1038/s41586-019-1116-4. https://doi.org/10.1038/s41586-019-1116-4

[3] P. Craievich *et al.*, "Consolidation and extension of the high-gradient linac RF technology at PSI," *Proc. LINAC'18*, pp. 937–940, 2019.

[4] R. Zennaro *et al.*, "High power tests of a prototype x-band accelerating structure for CLIC," *Proc. IPAC'17*, pp. 4318–4320, 2017.

[5] A. Solopova *et al.*, "SRF cavity fault classification using machine learning at CEBAF," in *10th Int. Particle Accelerator Conf.(IPAC'19), Melbourne, Australia, 19-24 May 2019*,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV021

JACOW Publishing, Geneva, Switzerland, 2019, pp. 1167–1170.

[6] C. Tennant, A. Carpenter, T. Powers, A. Shabalina Solopova, L. Vidyaratne, and K. Iftekharuddin, "Superconducting radio-frequency cavity fault classification using machine learning at Jefferson Laboratory," *Physical Review Accelerators and Beams*, vol. 23, no. 11, Nov. 2020, ISSN: 2469-9888. DOI: 10.1103/physrevaccelbeams.23.114601. http://dx.doi.org/10.1103/PhysRevAccelBeams.23.114601

[7] C. Obermair *et al.*, "Machine learning models for breakdown prediction in RF cavities for accelerators," *Proc. IPAC'21*, 2021.

[8] H. Kockelbergh and D. Dunning, "Summary of RF conditioning data analysis for cavity T24N5_L1 on Xbox 3," *STFC technical note*, 2020.

[9] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[10] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV022

SAMPLE ALIGNMENT IN NEUTRON SCATTERING EXPERIMENTS USING DEEP NEURAL NETWORK

A. Diaw *, K. Bruhwiler, J. P. Edelen, C. C. Hall, RadiaSoft LLC, Boulder, CO
S. Calder, C. Hoffmann, Oak Ridge National Laboratory, Oak Ridge, TN

Abstract

Neutron scattering facilities, such as Oak Ridge National Laboratory (ORNL) and the NIST Center for Neutron Research, provide a source of neutrons to investigate a wide variety of scientific and technologically relevant areas, including material science, chemistry, biology, physics, and engineering. In these experiments, the quality of the data collected is sensitive to sample and beam alignment. This alignment changes both between different samples and during measurements. The sample alignment optimization process requires human intervention to tune the beam and sample positions. While this procedure works, it is inefficient, time-consuming, and often not optimized to high precision alignment. This paper uses different neural network architectures on neutron camera images from the HB-2A powder diffractometer beamline at the High flux Isotope Reactor (HFIR) and optical camera images from the TOPAZ single crystal diffractometer at the Spallation Neutron Source (SNS), both at ORNL. The results show that the trained network on a few images accurately predicts the sample position on holdout test images. The resulting surrogate model can then be used via feedback-driven adaptive controls to tune the experimental parameters to get the desired beam and sample position.

INTRODUCTION

Neutron scattering experiments probe the structure and dynamics of materials to provide unique insights into physical, chemical, nanostructured, biological and engineering materials science. At present the US Government operates two premier neutron scattering user centers [1]; one at Oak Ridge National Laboratory and one at the NIST Center for Neutron Research. These large scale facilities offer flagship research capabilities to academia and industry that serve a broad and growing user community which necessitates their improved operational efficiency and capacity. The experiments involve collecting data on samples in a neutron beam on highly optimized neutron instruments. The position of the sample in the beam is often critical to the subsequent data analysis and measurements success, however the need to perform sample alignment and realignment during experiments reduces the amount of time spent collecting data. While there have been efforts to improve sample alignment [2] and to automate switching between samples [3, 4], manual intervention from facility beamline scientists or expert users is still necessary. At present optimization of the experimental environment requires a facility beamline scientist or expert user to interpret image data of the sample either through an optical camera or a neutron camera. A correction is then applied manually based on the data from these images until the sample is aligned in the beam. The automation of this process will result in more efficient and accurate use of neutron facilities and therefore increase the overall quality and reliability of their scientific output. During our studies we have focused on developments for two experimental stations, the HB-2A neutron powder diffractometer and the TOPAZ neutron single crystal diffractometer.

The HB-2A neutron powder diffractometer (as shown in Fig. 1) is located at the High Flux Isotope Reactor at ORNL and is primarily utilized for magnetic structure determination while focusing on experiments at ultra-low temperatures with the options of high external fields or applied pressures. The constant energy neutron beam in a low noise environment and a simple incident beam profile makes HB-2A well suited to a variety of interchangeable environments with minimal calibration required. There are a variety of sample changer options that can allow multiple samples to be loaded at a time. With a focus on weak magnetic signals the sample alignment and beam optimization with adjustable slits and sample stage motors is crucial to reducing unwanted background scattering and to maximise the sample signal. With further optimization of the beamline to include a new detector count rates would increase by over an order of magnitude, resulting in a significant increase in the number of samples being run, further motivating the advancement of automated sample alignment. For more details about the HB-2A instrument and scientific applications see [5].

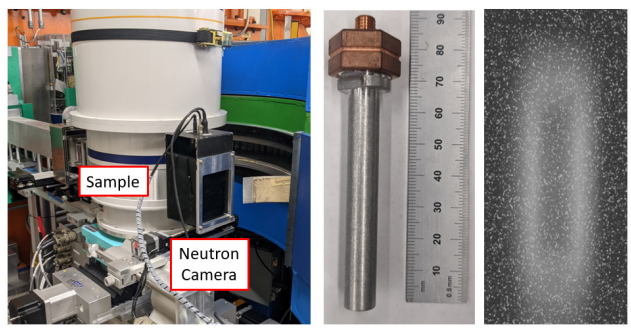


Figure 1: The HB2A powder neutron diffraction instrument. The sample and detector area is shown on the left, with the neutron camera behind the sample. Middle shows a standard sample holder that the powder sample is loaded into, with the scale shown in millimeters. An example of a neutron image is shown on the right, with the powder appearing as a shadow-like image in the white beam.

* diaw@radiasoft.net

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV022

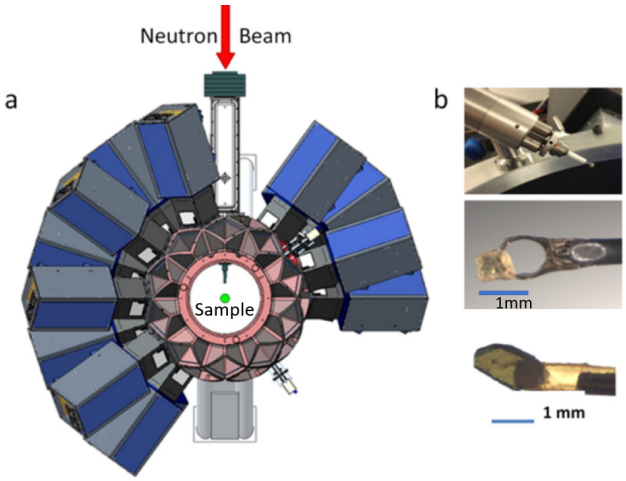


Figure 2: TOPAZ neutron single crystal diffraction instrument. (a) The sample and detector area is shown as viewed from above the instrument. (b) Top image shows the sample goniometer arm. Samples are loaded on pins and attached to the end of the arm. The middle and bottom images show images of the samples attached to sample mounts.

The TOPAZ neutron single crystal diffractometer (as shown in Fig. 2) is located at the Spallation Neutron Source at ORNL and offers high resolution measurements of crystal structures in small sample volumes. The smallest crystals studied to date in this instrument are 0.065 mm^3 in size. TOPAZ has recently been upgraded to include the installation of a focusing high-flux neutron guide end-section. This enables routine measurements of sub-millimeter crystal samples. TOPAZ is also capable of continuous 3DQ-space mapping for a specific region of reciprocal space volume given a stationary single-crystal sample. This is particularly useful when studying phase transitions as a function of temperature or other external stimuli. TOPAZ possess state of the art data acquisition and reduction tools making it one of the more advanced neutron scattering end stations available for users. These capabilities, however, are under utilized in part due to limited temperature range and manual alignment of the sample in the environment. Improved tools for automatic sample alignment and stabilization promise to increase the throughput of the detector and therefore increase its net scientific output. For more specifics about the TOPAZ diffractometer and other diffractometers at ORNL see [6].

The two neutron beamlines HB-2A and TOPAZ use different methods to image the sample and optimize the alignment. HB-2A uses a neutron camera that is placed in the neutron beam after the powder sample. The sample and any ancillary equipment that are in the neutron beam appear as a shadow in the beam, with the contrast varying depending on the neutron absorbing characteristics of the material. The beam size has a maximum of 20 mm × 60 mm (Horizontal × Vertical) and the samples are typically enclosed within a cylindrical can ~ 60 mm tall with diameters in the range 4–15 mm. The target sample alignment is to center the sample in the beam horizontally and vertically, translate to the center of the detector arc to ensure reliable data quality and to mask out any unwanted beam not hitting the sample that would add unwanted background scattering. For the TOPAZ diffractometer, centering of samples in the neutron beam is accomplished via an optical camera. The sample sizes are in the mm to sub mm size, with a beamsize of 2.0–4.0 mm in diameter. The sample shape and dimension are often irregular and can vary significantly between samples. To view the sample a video camera is installed below the sample and individual images or series of images with respect to a marked beam center position are recorded. The sample is aligned remotely using a software algorithm. All associated metadata and beamline configuration data are recorded and associated with the image. This alignment procedure is done both at the start of the experiment when a sample is initially installed and after the sample temperature has changed which results in a non-trivial alteration of the sample position due to thermal expansion/contraction of the sample environment. Temperature changes can occur multiple times per experiment.

Machine learning has recently been of interest for a range of automation tasks in particle accelerators [7, 8] and experimental end stations [9, 10]. Convolutional neural networks are of particular interest due to their ability to efficiently process image data [11, 12]. Here we will detail the prototyping and testing of new machine learning methods for sample alignment and stabilization of samples at neutron scattering beamlines. Our approach employed the use of convolutional neural networks to either relate diagnostic images to a series of machine settings in operational experiments or reconstruct a binary image outlining the sample. These algorithms can then be used during neutron experiments in conjuction with a feedback algorithm to stabilize samples in the beam. This paper provides an overview of our recent efforts to build convolutional neural networks to assist with the automation of sample alignment in neutron beamlines.

METHOD AND RESULTS

We focus on designing an efficient approach to dealing with images of different samples obtained during the neutron-scattering experiment.

First, we consider the HB2A dataset. Figure 1 shows an example of the neutron camera image with the powder appearing as a shadow-like image in the white beam. Our dataset

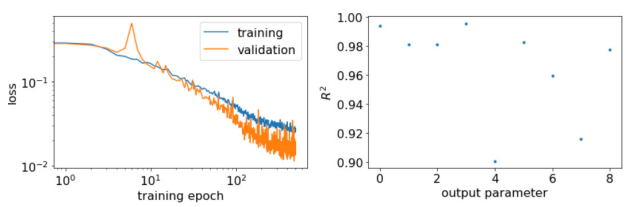


Figure 3: Left: Training an validation loss function at each epoch. Right: the coefficient of determination R^2 for each of the 9 motor parameters.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV022

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

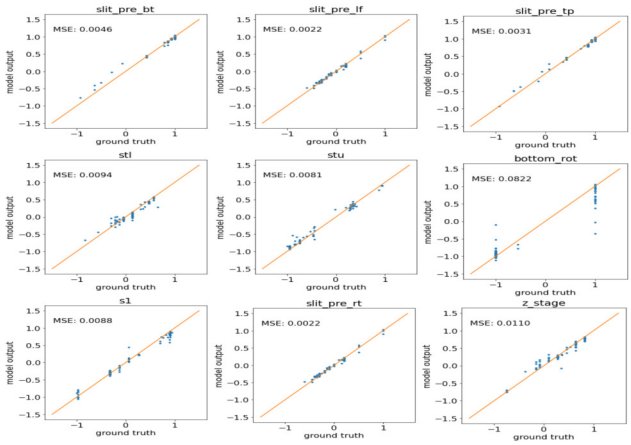


Figure 4: Test of the CNN surrogate performance in the test dataset. Each plot title corresponds to a different instrument motor that can control sample position and beam size. On the x-axis we have the data from the experiment, y-axis represents the model prediction.

consisted of a collection of 530 images sized $480 \times 640 \times 4$ and 9 relevant motor parameters such as *smpl_stk* to selecting the motor position for a sample, *stu* to translate the sample horizontally in the beam or *slit_pre_tp* to control the beam size with motorized slits. Our focus was to find a surrogate model that can take a neutron image and predict the motor parameters. This was accomplished using vanilla convolutional neural networks implemented in keras [14]. These structures are well suited for extracting scalar quantities from image data. The CNN consists of a series of 4×4 convolution layers. Each layer comprises three types of operations: convolution, rectified linear units, and batch normalization. A dropout of 0.25 is applied between layers and there is a dense layer (256) after the last convolution. A network was trained to learn the behavior of a dataset; this means fine-tuning each layer's parameters (weights and biases) to produce a valid model. We employ a conventional formulation based on the minimization of a loss function L with the mean-squared error:

$$L = \frac{1}{n_{\text{batch}}} \sum_{s,t} (\hat{X}_s^t - X_s^t)^2. \tag{1}$$

Here s loops over the training examples, and t runs over training targets. $\hat{X}_s^t$ is the t-th image for example s, and X_s^t is the output of the t-th neuron in the last layer of the network, and where n_{batch} is the batch number. We minimize this output by accelerated gradient descent with Adam optimizer with a learning rate set to $lr = 10^{-4}$ in a series of *epochs*.

Figures 3 and 4 illustrates the network performance. For each motor position, the coefficient of determination $R^2 > 0.9$, indicating that the training approach yields robust surrogate models.

Although we successfully trained a network on the dataset, the inverse problem of predicting a sample position from the CNN model using the machine setting parameters did not yield good results. Much of this is related to the degenerate

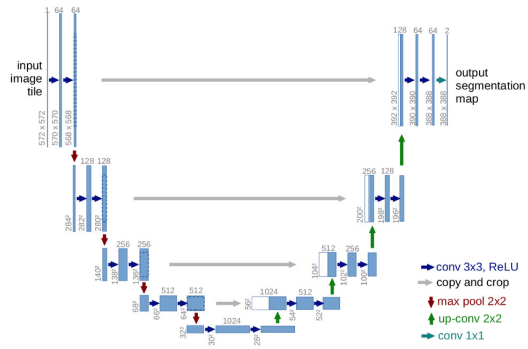


Figure 5: The proposed U-net architecture uses a contraction and an expansion paths. The blue boxes represent multi-channel feature map while the white boxes are their copies. The gray arrows correspond to the operations from left to right. This figure was taken from Ref. [13].

nature of the system. This limits our ability to provide a feed-forward adjustment to the motors based on a neural network model. An alternative approach is to identify the sample position from the image and use feedback on the motors to position the sample. This sample identification approach was applied to the Topaz dataset using a slightly different architecture.

For the Topaz data, our problem is as follows: Given an image, localize sample position in the image and return the center of the mass of the sample. To do so, we trained a convolution neural network based on the segmentation algorithm 2D-U-net [13]. The U-net architecture shown in Fig. 5 consists of contracting and expansive paths and overall has 23 convolutional layers. For the contracting layer, a vector input y is passed to two 3x3 convolutions, each followed by an activation unit, and produces a vector of activations, y', via

$$y' = f(Wy + b), \tag{2}$$

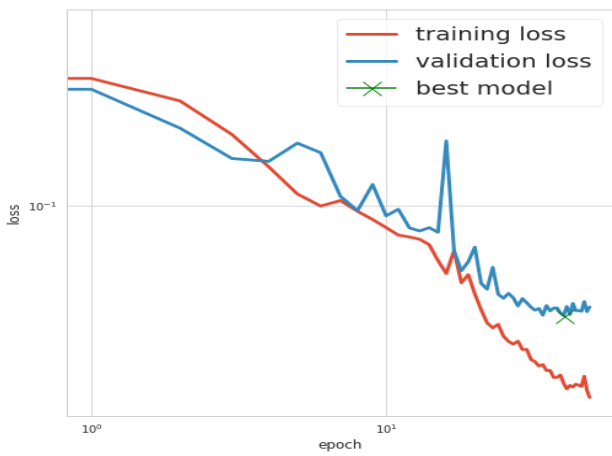


Figure 6: Learning curve. Training and validation loss at each *epoch*.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV022

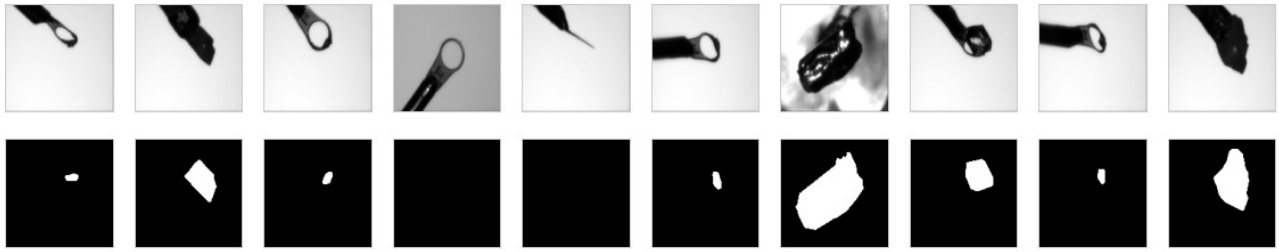


Figure 7: Examples images obtained from the experiment. The lower plots show binary image masks created around the sample. For images with no sample the mask is empty.

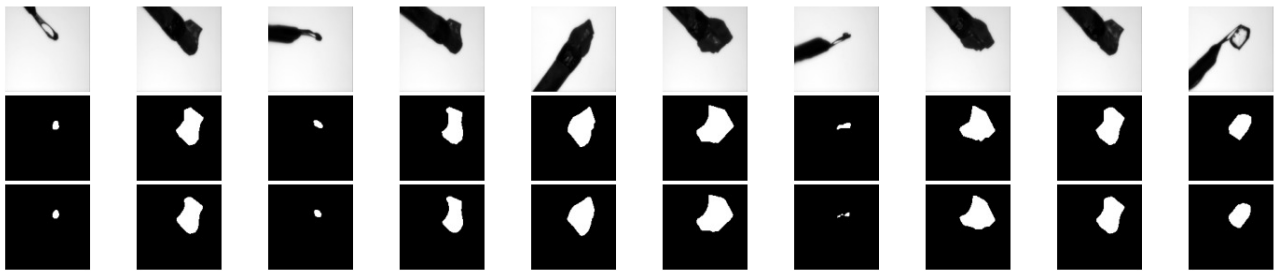


Figure 8: Test model performance for some random images in the test dataset. Top row are the test image (ground truth), middle row: the test image label, and bottow row are the predicted test binary by the U-net algorithm. The predicted test image is visually identical to the ground truth.

where W is a weight matrix, b is a bias vector, and f is an activation function. We choose the rectified linear unit (ReLU) function $f(s) = \max(0, s)$. Then, we introduce a downsampling layer consisting of 2×2 max pooling operation with stride 2. The downsampling factor was set 2 at each step while the number of feature channels doubles. For the expansive path, we have an upsampling operator of the feature map. A 2×2 convolution is employed to reduce the number of feature channels and two 3×3 convolutions. After the last convolution layer, we applied a 1×1 convolution to map each 64-component feature vector to produce the desired clean image.

Our training dataset is composed of images and the corresponding masks (binary), where each image of size 964×1292 has either a sample or no sample, as shown in Fig. 7. Because these images are extremely large, we resized them to 128×128 before applying them to the network. Although this procedure may introduce artifacts in the sample images, it did not seem to impact the result of the training. The dataset consists of 611 images, and it was obtained by randomly varying the parameter settings of the optical camera images from the TOPAZ single crystal diffractometer. All model evaluation was carried out using a hold-out 20% validation set, which contains no overlap with the training set. We used a formulation based on the cross-entropy loss function. This latter is minimized by accelerated gradient descent with the Adam optimizer in a series of epochs, that is, passes over the entire training dataset, using a learning rate of 10^{-3} and batches of size $n_{\text{batch}} = 32$. A patience-based learning scheduler is used to stop the training and revert to the last best network if the validation loss does not improve over $n_{\text{patience}} = 10$ *epochs*, with a global termination of $n_{\text{epochs}} = 500$ at maximum. Figure 6 illustrates the training and validation loss error. The network terminates within few epochs (~ 50).

Finally, Fig. 8 shows random samples from the ground truth data, their labels, and corresponding predictions obtained using our surrogate model, demonstrating that the u-net approach captures details very accurately across the sample types.

CONCLUSION

Machine learning tools are promising for the increased automation of neutron beamlines. We have shown the ability for convolutional neural networks to predict motor positions from images in the HB-2A beamline and to contour the sample for optical images collected from the TOPAZ beamline. Using the output of the U-net architecture we can compute the center of mass of the sample and hand that off to the motor controllers in order to maintain sample alignment during the experiment.

ACKNOWLEDGMENTS

This work was supported by the DOE Office of Science Office of Basic Energy Science SBIR award number DE-SC0021555. This research used resources at the High Flux Isotope Reactor and Spallation Neutron Source, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV022

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] DoE, "Neutron Scattering Facilities: U.S. DOE Office of Science (SC)", July 2019.

[2] J. A. O'Toole, "Advances in neutron science instrumentation at the Los Alamos Neutron Science Center (LANSCE)", *IEEE Nuclear Science Symposium Conference Record*. vol. 1, pp. 627–632, 2007.

[3] H. M. Reiche and S. C. Vogel, "A versatile automated sample changer for texture measurements on the high pressure-preferred orientation neutron diffractometer", *Review of Scientific Instruments*, vol. 81, no. 9, p. 093302, 2010. doi:10.1063/1.3485035

[4] J. E. Rix *et al.*, "Automated sample exchange and tracking system for neutron research at cryogenic temperatures", *Review of Scientific Instruments*, vol. 78, no. 1, p. 013907, 2007. https://doi.org/10.1063/1.2426878

[5] S. Calder *et al.*, "A suite-level review of the neutron powder diffraction instruments at Oak Ridge National Laboratory", *Review of Scientific Instruments*, vol. 89, no. 9, p. 092701, 2018. doi:10.1063/1.5033906

[6] L. Coates et al., "A suite-level review of the neutron single-crystal diffraction instruments at Oak Ridge National Laboratory", *Review of Scientific Instruments* vol. 89, no. 9, p. 092802, 2018. doi:10.1063/1.5030896

[7] A. L. Edelen *et al.*, "Neural Networks for Modeling and Control of Particle Accelerators", *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 878–897, Apr. 2016. issn: 0018-9499.

[8] A. L. Edelen *et al.*, "Opportunities in Machine Learning for Particle Accelerators", 2018. arXiv:1811.03172

[9] S. C. Leemann *et al.*, "Demonstration of Machine Learning-Based Model-Independent Stabilization of Source Properties in Synchrotron Light Sources", *Phys. Rev. Lett.*, vol. 123, p. 194801, 19 Nov. 2019.

[10] B. Nash *et al.*, "Reduced model representation of x-ray transport suitable for beamline control", in *Proc. SPIE 11493, Advances in Computational Methods for X-Ray Optics V, (SPIE Proceedings Vol. 11493)*, 21 August 2020, pp. 53–61. doi:10.1117/12.2568187

[11] R. Venkatesan and B. Li., *Convolutional Neural Networks in Visual Computing: A Concise Guide*, CRC Press, 2018.

[12] B. Nash *et al.*, *Convolutional Neural Networks in Python: Beginner's Guide to Convolutional Neural Networks in Python*, CreateSpace Independent Publishing Platform, 2018.

[13] O. Ronneberger *et al.*, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015. arXiv:1505.04597

[14] F. Chollet *et al.*, "Keras", 2015. https://github.com/fchollet/keras

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

DEVELOPMENT OF A SMART ALARM SYSTEM FOR THE CEBAF INJECTOR*

Dan Tyler Abell, Jonathan Edelen, RadiaSoft LLC, Boulder, Colorado, USA
Daniel Grady Moser, Brian Freeman, Reza Kazimi, Chris Tennant
Thomas Jefferson National Accelerator Facility, Newport News, Virginia, USA

Abstract

RadiaSoft and Jefferson Laboratory are working together to develop a machine-learning-based smart alarm system for the CEBAF injector. Because of the injector's large number of parameters and possible fault scenarios, it is highly desirable to have an autonomous alarm system that can quickly identify and diagnose unusual machine states. We present our work on artificial neural networks designed to identify such undesirable machine states. Our initial efforts have been focused on model prototyping and data curation. In this paper we present an overview of our initial findings and our efforts to generate a robust dataset for the alarm system. We conclude with a discussion of plans for future work.

INTRODUCTION

A significant aspect of accelerator operations involves identifying the root causes of faulty machine states. For example, the machine trips on a beam loss monitor. But why? What underlying cause trips the machine? Sometimes the reason is obvious, while other times not. Existing alarm systems commonly indicate when specific machine parameters drift outside their normal tolerances. However, operators must still interpret these alarms in the context of many interacting systems and subsystems before they can take the the most appropriate corrective action.

The project described in this paper has at its primary objective the development of a machine-learning-based model with the ability to rapidly identify potential root causes of machine faults (hence the term Smart Alarm). More specifically, given machine readings (defined more precisely later), the system continuously compares the model's predictions of the expected machine settings (also defined later) against actual machine settings. When a discrepancy arises that exceeds some user-defined threshold, the system raises an alarm that directs operators (or subject matter experts) to the "bad" setting (*e.g.*, corrector, solenoid, rf gradient, *etc.*). If this effort succeeds, a more ambitious goal will be to extend the work to monitor the machine for parameter drifts and identify when the machine needs "tweaking" before a fault event occurs.

During our initial efforts at training and validating machine learning (ML) models, we obtained puzzling and disappointing results. The problems at issue we traced back to various difficulties with our data, including some outlier (read nonsensical) vacuum readings. In the process of that investigation, we examined our data more carefully and found other aspects of data collection and selection that required careful consideration. The lesson to learn here is that *understanding one's data*—though time-consuming—*constitutes a critical part of any ML effort.*

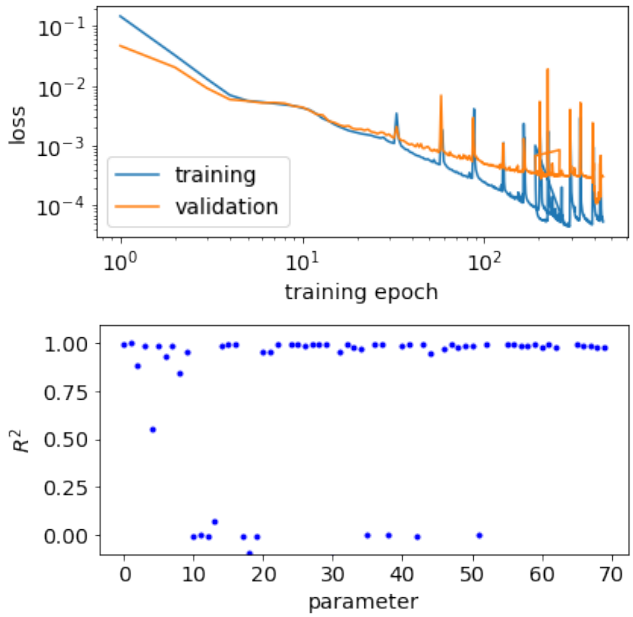


Figure 1: Inverse model trained on operations data from the JLab injector.

In the following sections, we describe briefly some of our early ML efforts and how they led us to make a thorough investigation of our data and the data collection process. We then describe our data, the collection process, and our evolving understanding of how best to curate data that will prove useful for the training and validation of future ML models. We conclude with our plans for future work.

INITIAL ML EFFORTS

Among our first efforts was training an inverse model on data taken from the JLab injector, with the goal of using measurements (readings) to predict machine settings. Figure 1 shows the loss functions and the somewhat disappointing R^2 fit results for one training run with this data. Modifications to the network architecture, number of epochs, choice of optimizer, *etc.*, made relatively little improvement on the overall results.

This circumstance led us to undertake a critical examination of our data, which at the time comprised that taken during (i) regular *operations* of the injector and (ii) a dedicated *study* of the machine. The various parameters (called

* Work supported, in part, by the US Department of Energy, Office of Science, Office of Nuclear Physics, including grant No. DE-SC0019682.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

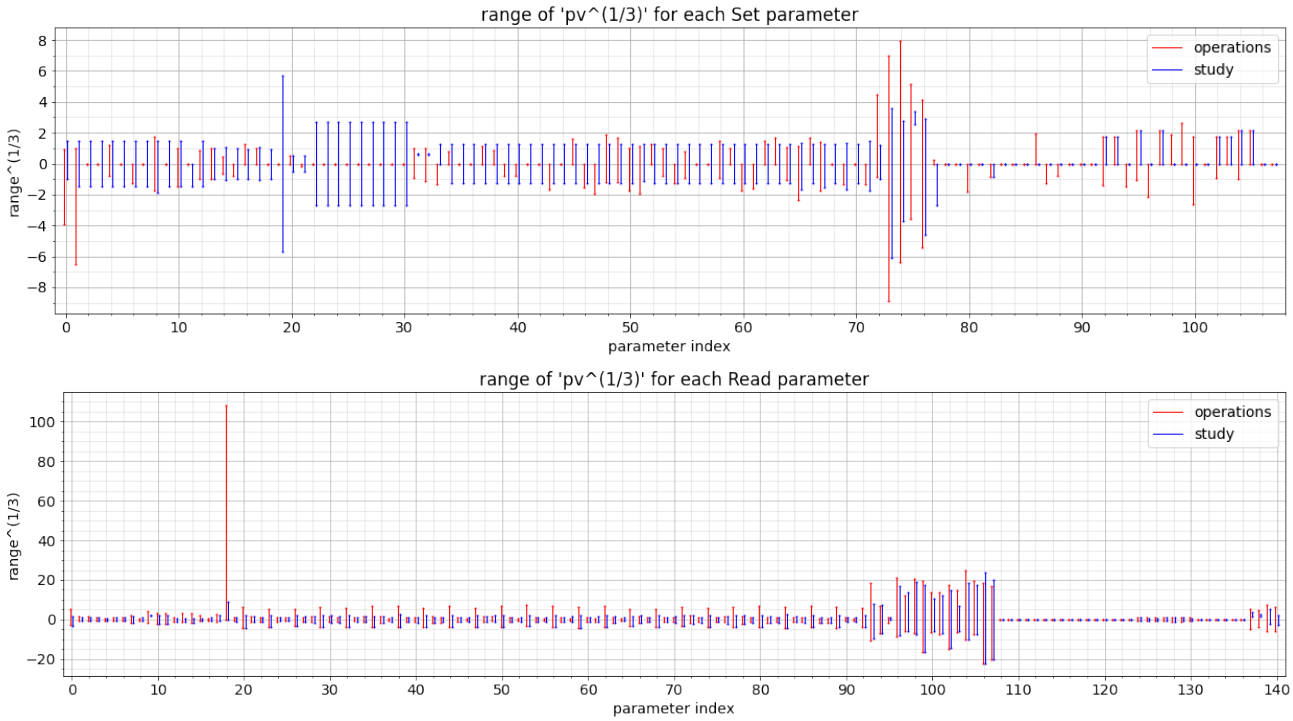


Figure 2: Summary of all Set (upper) and Read (lower) PV data. For each index, the red and blue vertical bars show the range of values spanned respectively by injector operations data and by injector study data. The vertical scales have been compressed by taking a cube-root. The medians of the operations data have been subtracted out.

Process Variables, or PVs) are divided into *Settings* and *Readings*. Figure 2 shows the ranges spanned by the data taken for each of the PVs, and in particular the overlap of the ranges between operations and study data. The hope was to use, say, operations data (of which we had much more) for training and validation, and the study data for testing. For that to work well, however, one would need to see that, say, the range of each *Study* PV lies within the range of its corresponding *Operations* PV—or vice versa. But, as shown by the graphics in Fig. 2, neither case consistently holds.

Another aspect of the data, not illustrated here, is that many of the Set PVs take on just a very few discrete values. This minimal variation suggests that it will, as we have found so far, prove difficult for ML networks trained on this data to achieve a useful characterization of the JLab injector. The solution, of course, is more data and, more importantly, better data.

A newer dataset from the JLab injector comprises some 409 500 instances averaged over one-second intervals for some seven weeks last summer. (See the next section for more details.) This data has no natural divide along which one might assign the different records into separate train and validation datasets; hence we attempted such a split randomly. In this case, too, we found some PVs for which the training range covers the validation range, and other PVs for which the reverse holds. This presents a challenge for training because there are ranges where the model would be trying to extrapolate to a new domain. This, a fundamental challenge for machine learning, and one of the challenges of model-based anomaly detection, means that a full understanding of the data is critical to success.

CURATING THE DATA

Five datasets were collected using several different methods. The variety of approaches reflects how our understanding of capturing machine data has evolved (and how it continues to do so). The results of this work will likely inform the modification of current—or the development of new—tools to capture and/or mine data more efficiently.

Collected Datasets

The datasets collected exist in tabular form, with each column representing an EPICS process variable (PV), and each row a particular snapshot in time. For example, collecting data for the three PVs `Date`, `R047GSET`, and `R047PSET` every hour, on the hour, for two days of operation would yield a dataset of size of 48 × 3 [= (2 days × 24 readings/day) × (3 PVs)] (as shown in Fig. 3). We also give the range of dates (to the nearest day) covered by each dataset below (listed in the order collected).

- **Dataset A** Data collected from dedicated beam studies in which select injector beamline components were systematically varied and downstream responses recorded. Date: 2020-09-08 (swing shift) and 2020-09-14 (swing shift). Size: 654 × 223.
- **Dataset B** First attempt at mining the operational archiver for "good beam". Date: 2020-08-30 to 2020-09-18. Size: 24 × 302.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

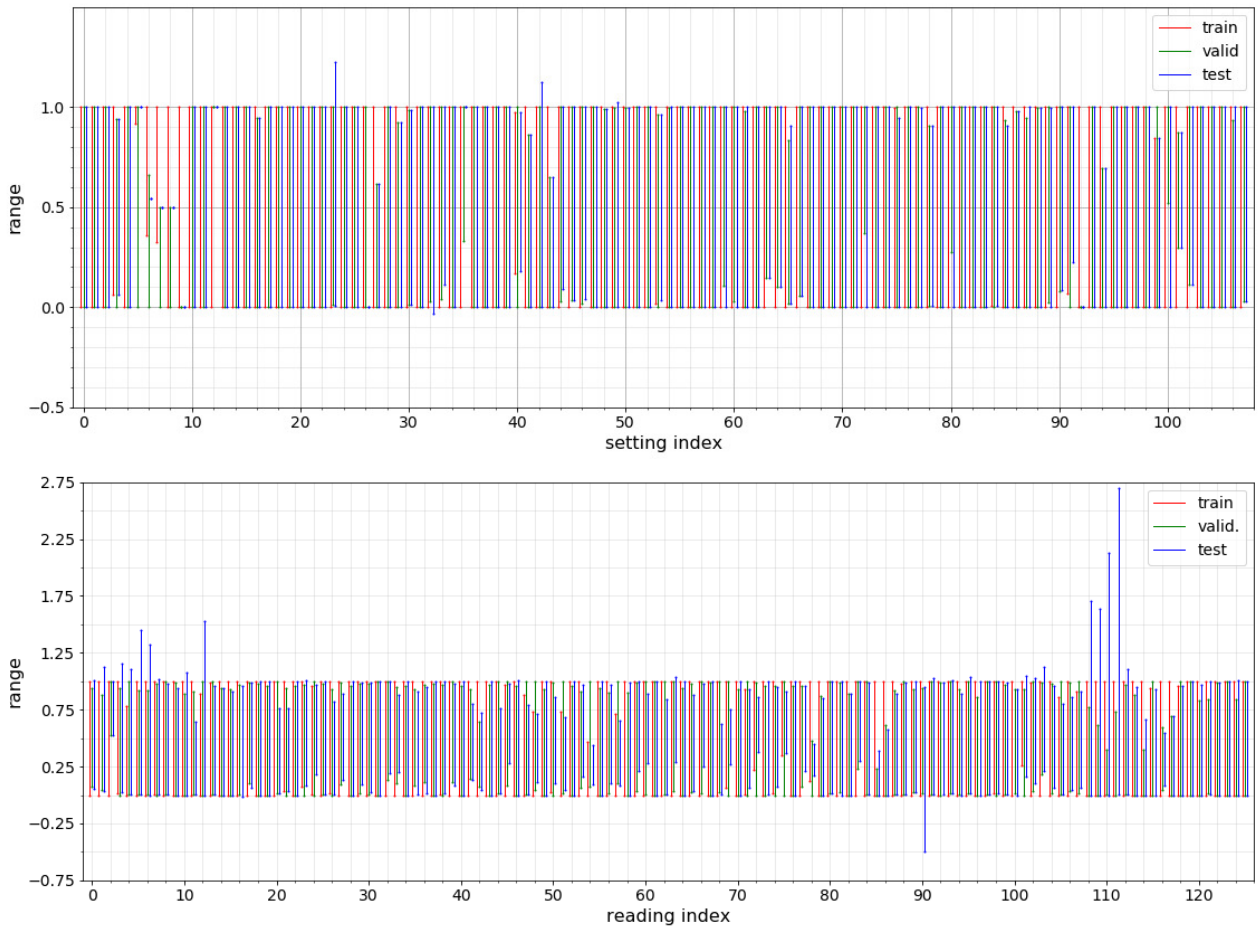


Figure 3: Ranges of the Scaled setting (upper) and Reading (lower) PVs, color coded by data split.

- **Dataset C** Mining the operational archiver for "good beam" (using less stringent constraints). Date: 2020-08-01 to 2020-09-21. Size: 5467 × 303.
- **Dataset D** Mining the operational archiver without imposing "good beam" constraints. Data was collected and averaged over each second of the Fall 2020 operational run. Date: 2020-08-01 to 2020-09-20. Size: 409 549 × 318. (This represents an initial filtering for beam-on conditions.)
- **Dataset E** Mining the historical archiver without imposing "good beam" constraints. Data was collected hourly for the last two years. Whereas the operational archiver stores machine data from the last several months, the historical archiver contains data from the previous few years. Date: 2019-02-01 to 2021-01-30. Size: 17 520 × 262. (The period from 2020-09-20 to 2021-01-30 represents a scheduled accelerator down period and contains no useful data.)

Several observations can be made. First, many of the datasets contain overlapping dates. Second, each dataset records a different number of PVs. Finally, there is a clear evolution in data collection strategy: It started with conventional beam studies efforts where knobs were varied and responses recorded (Dataset A). While this provided several hundred examples, both the complexity of the problem we are trying to solve and the means with which to solve it (deep learning) necessitates *much* more data. We then transitioned to mining the archiver for PV values. Initial efforts placed the burden on carefully designing constraints so as to ensure that we collected only PVs corresponding to "good beam". This resulted in several thousand examples (Dataset C). We then transitioned to a mode of a collecting data from the archiver at regular intervals (either hourly or every second) with no filtering. This shifted the burden of extracting the appropriate data for training models from the data collection effort to the data pre-processing step.

Cleaning the Data

Because the dates of Dataset D overlap those the three previous datasets (A, B, and C), we now focus only on Datasets D and E. Some initial "cleaning" of the data involved clerical matters—reformatting, renaming columns, combining and the like. The more significant matters included various filters applied to the data:

- Refine filtering from "beam on" to "beam being on" and transported to the end of the injector.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV023

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Remove those (few) columns that contain a large number of NaNs.
- If a column contains a missing value (*e.g.*, NaN), remove the associated row (across all columns).
- Some PVs take only integer values (*e.g.*, 0 or 1). Because of the averaging the entries occasionally have non-integer values, indicating a change of state. Remove those rows.

After combining the resulting versions of Datasets D and E, removing duplicate rows, and retaining only the common set of PVs (there are 235), we arrive at a dataset of size $406\,397 \times 235$.

Partitioning the Data

Regardless of neural network architecture, one needs a well-defined set of inputs that the model learns to map to a well-defined set of outputs. Our next step here involves partitioning the 234 PVs (we exclude the Date column) into appropriate Setting and Reading datasets. At first, we define a *Setting* as any PV that an operator can adjust during routine beam tuning. These include phase and gradient set-points of radio-frequency (rf) cavities, solenoid strengths, corrector strengths, and the like. A *Reading*, on the other hand, we characterize as read-backs of various diagnostic systems. These include readings from beam loss monitors (BLMs), beam position monitors (BPMs), vacuum signals, beam current monitors (BCMs), and statistical descriptions of the beam as extracted from a synchrotron light monitor (SLM) image. In the end, we identify 108 Setting and 126 Reading PVs.

Executing a partition, however, proves less obvious than it sounds. Consider the beam current monitors: Should they count as Readings, as seems obvious, or as Settings? The BPM wire-sum signals, clearly Readings, act as a proxy for the beam current, which would seem to be a Setting. Depending on the details of a given ML architecture, we might want the model to learn the BCM current readings from BPM wire sums, or learn the wire sums from the BCM readings. In either of these cases, we shall have to reclassify some PVs from Readings to Settings. Or perhaps we ought to add laser power PVs as Settings, so as to capture an alternate proxy for the beam current treated as a Setting.

In the course of this review of PVs, we identified other issues: Several PVs do not appear in the data, including settings for one of the rf cavities and readings from several vacuum PVs. In addition, solenoids and correctors appear in the data, but quadrupoles do not. With regard to excluding he quadrupole PVs, the initial thinking was that quadrupoles do not often change value, and static (or nearly static) training data contains very little not serve to teach a given ML model. Whether to include then in the future requires further consideration.

For future data collection efforts, each injector PV need to be examined and a decision made as to whether or not to include it in the dataset for addressing the problem at hand. Moreover, those PVs should be explicitly defined at the outset, so as to ensure that all subsequent data collection efforts include a common, consistent set of PVs.

FUTURE WORK

Two primary challenges underlie the construction of a Smart Alarm system. The first is feature selection/down-selection, and the second is choosing an appropriate model. We will explore the use of different feature selection tools to develop reduced representations of given datasets. In addition to auto-encoders,we plan to develop both linear and nonlinear classifiers, including the possible use of neural networks and decision trees as nonlinear classifiers. We will compare the performance of different classifiers trained using supervised learning and establish some general heuristics for fault detection on this type of injector. We will begin by identifying a few simple fault scenarios, and then build on this to include a range of different faults, including those caused by coupled errors within the machine. We will also explore the use of unsupervised learning to train classifiers. Algorithms such as DBSCAN, gaussian mixture modeling, and agglomerative clustering are well suited for clustering a variety of distinct types of data. Once trained, those models can track the vicinity of a given machine state to the locations of distinct clusters determined from training.

CONCLUSION

Data provides the fuel for machine learning. In the context of the CEBAF injector, we are still learning how best to collect and curate the appropriate data for training our models. The archiver provides a rich—though largely underutilized—source of data. Using the toolkit of data science, we believe there is potential for a deeper understanding of machine performance from data that has already been collected.

ACKNOWLEDGMENTS

This work is supported, in part, by the US Department of Energy, Office of Science, Office of Nuclear Physics, including grant No. DE-SC0019682.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV024

X-RAY BEAMLINE CONTROL WITH MACHINE LEARNING AND AN ONLINE MODEL*

Boaz Nash, Dan Tyler Abell, David Leslie Bruhwiler, Evan Carlin, Jonathan Edelen, Michael Keilman, Paul Moeller, Robert Nagler, Ilya V. Pogorelov, Stephen Davis Webb
RadiaSoft LLC, Boulder, Colorado, USA
Maksim Rakitin, Yonghua Du, Abigail Giles, Joshua Lynch, Jennefer Maldonado, Andrew Walter
Brookhaven National Laboratory, Upton, New York, USA

Abstract

We present recent developments on control of x-ray beamlines for synchrotron light sources. Effective models of the x-ray transport are updated based on diagnostics data, and take the form of simplified physics models as well as learned models from scanning over mirror and slit configurations. We are developing this approach to beamline control in collaboration with several beamlines at the NSLS-II. By connecting our online models to the Blue-Sky framework, we enable a convenient interface between the operating machine and the model that may be applied to beamlines at multiple facilities involved in this collaborative software development.

INTRODUCTION

Electron storage ring based light sources and x-ray FELs provide radiation for a vast array of scientific research. The radiation is created in either undulator or bending magnet sources and then passes down an optical beamline, transporting the radiation to the sample for use in a scientific study.

Although modeling of beamline components and x-ray transport plays an important role in the design and commisioning of x-ray beamlines, these methods are largely unused during day to day operations. In comparison to the electron beam storage ring, in which reduced models combined with diagnostics play a crucial role in electron beam control, little has been done on the x-ray beamline side regarding such methods.

Modeling codes for x-ray beamlines can generally be broken into two classes, either wavefront propagation or ray tracing. The two most widely used codes of these types are Synchrotron Radiation Workshop (SRW) and Shadow. Whereas in principle, either of these codes could be suitable for an online model, there are some deficiencies which we look to improve. In particular, ray tracing methods, though fast and accurate, do not take into account diffraction effects or the effects of partial coherence. Wavefront propagation, on the other hand, while including diffraction effects, is substantially more computationally intensive, particularly when partially coherent computations are required.

We propose two types of simplified models that provide additional tools with which to build fast online models for x-ray beamlines. The first is a physics based model based on what we call a *Matrix-Aperture Beamline*[1]. And the second type of model is a surrogate model using the methods of machine learning on either simulated or measured training data. We describe our progress on the development of each of these types of models and their application to x-ray beamlines at NSLS-II at Brookhaven National Laboratory.

Next, in order to create an accurate online model that ties the beamline and photon beam status to a software model, we must make measurements at beamline diagnostics and take into account beamline component positions using the existing beamline control software. For this purpose, we use the BlueSky software, developed at NSLS-II, but with a goal for wider adoption across the world of synchrotron light sources in the US and beyond.

We document here our continuing efforts to improve this situation by creation of online models in synchrotron light sources that may be used for automated beamline control. We are working with the TES bending magnet beamline at NSLS-II [2], and thus draw our simulations and control examples from this beamline.

REDUCED PHYSICS MODELS

Our reduced physics models are based on the concept of a *Matrix-Aperture-Beamline*. We call the code we are building to implement this concept `MABTrack`, and a diagram for the case of two apertures is shown in Fig. 1. The `MABTrack` code is being developed on GitHub in the `rslight` repository[1] and will be available as an open source package.

The `MABTrack` code starts with a model in SHADOW to compute reference orbit $\vec{Z}_0(s)$, ABCD matrices $M_j(s)$ and physical apertures $t_j(\vec{x})$. Three levels of sophistication in modeling are being developed within `MABTrack`. At the simplest level, one computes the propagation of the radiation second moment matrix $\Sigma(s)$, defined as $\Sigma_{jk}(s) = \langle \vec{\tilde{z}}_j \vec{\tilde{z}}_k \rangle$ with $\vec{\tilde{z}}(s) = \vec{z}(s) - \vec{z}_0(s)$, i.e. subtracting off the reference orbit. The second moments propagate as

$$\Sigma(s) = M(s)\Sigma_0 M^T(s) \tag{1}$$

with Σ_0 as the initial values of the second moments.

As a second level of sophistication, we will use *linear canonical transforms* (LCTs) to transport wavefronts [3]. LCTs have an intimate connection to the so-called *ABCD*

* Work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, including grant No. DE-SC0020593

[1] `https://github.com/radiasoft/rslight`

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV024

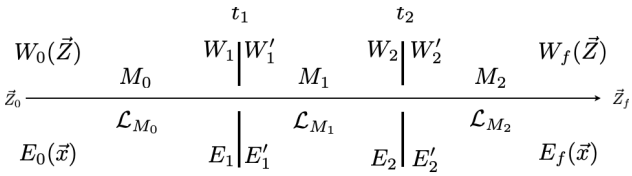


Figure 1: Structure for `MABTrack`, Matrix Aperture Beamline Tracking code for the case with 2 physical apertures $t_1(\vec{x})$ and $t_2(\vec{x})$. Transfer matrices (M_0, M_1, and M_2) are assumed to be known, either analytically, or via a ray tracing code such as Shadow. Either electric fields $E_0(\vec{x})$ or Wigner functions $W_0(\vec{Z})$ may be propagated through the beamline.

matrices of ray optics, and constitute a generalization of the various integral transforms—including Fresnel transforms, Fourier transforms, fractional Fourier transforms, scaling, and chirp multiplication (multiplication by a Gaussian)—used for computations in wave optics. In one degree of freedom, one may write the LCT of a function g in the form [3]

$$\tilde{g}^M(u) = \mathrm{e}^{-\mathrm{i}\pi/4}\sqrt{\beta} \times \int_{\infty}^{\infty} \exp[\mathrm{i}\pi(\alpha u^2 - 2\beta u u' + \gamma u'^2)] g(u')\,\mathrm{d}u'. \quad (2)$$

As an alternative to the parameters α, β, γ, one may instead parameterize 1D LCTs in terms of the entries of a 2×2 symplectic matrix M—an *ABCD* matrix, the superscript seen on the left-hand side of (2)—according to the rule

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \gamma/\beta & 1/\beta \\ -\beta + \alpha\gamma/\beta & \alpha/\beta \end{pmatrix}. \quad (3)$$

When computed naïvely, the integral transform (2) represents a computationally expensive task. To address this difficullty, one may take advantage of the *group property* of the LCT, which states that

$$\tilde{g}^{M_2 M_1} = \tilde{g}^{M_2} \circ \tilde{g}^{M_1}. \quad (4)$$

This property tells us that if one can factor the matrix M into a product of (simpler) matrices M_1 and M_2, and if one can easily compute the LCTs corresponding to those simpler matrices, then one may easily compute the full LCT by composition. This remains true even if when we factor M into a product of more than two matrices.

In special cases, the LCT becomes quite tractable. In the case $\alpha = \gamma = 0$ and $\beta = 1$, for example, the LCT becomes, apart from a normalization, a Fourier transform. Other choices of the parameters lead to the other well-known transforms mentioned above, for which fast algorithms are known. Moreover, it is always possible to factor M into a product of matrices corresponding to such transformations. Indeed, there exists a substantial literature on this topic. See, for example, [4–10]. It is worth mentioning that, as described in some of the references, that the general procedure described above works also for 2D LCTs. In addition, we mention that, despite the large literature, it appears that no (publicly available) LCT library currently exists. Hence the need to write one of our own.

To implement the LCT, we have begun with the case of one degree of freedom, and have implemented (i) some needed utility functions, (ii) the component LCT functions, and (iii) functions to decompose the given LCT into simpler parts and to compose the resulting component LCTs. Because this code is being written in Python, we make—in writing all these functions—extensive use of `numpy`[2], a high-performance numerical library for Python.

The mentionad utilities enable us to convert between parameters a, b, c, d and α, β, γ (`convert_params_3to4` and `convert_params_4to3`); construct the abscissae for a given signal (`abscissae`); and resample a given signal (`resample_data`), which is needed because chirp multiplication increases the effective time-bandwidth product [5, 9]. All these utilities have been implemented and tested.

For the component LCTs, we need scaling (`scale_data`), the Fourier transform (`fourier`), and chirp multiplication (`chirp_multiply`). All three of these have been written, and we are in the process of testing.

The remaining functions we need are one for decomposing a given LCT into simpler parts (`lct_decompose`), and a second for composing the component LCTs (`apply_lct`). These functions have also been written, and they have passed some (rather trivial) tests. Final testing will happen after we complete testing of the component LCTs. When the LCT algorithms have been tested, they will be integrated into `MABTrack`. This effort will then be generalized to 2D LCTs.

The final level of sophistication in `MABTrack` involves direct propagation of Wigner functions which allows for inclusion of partial coherence. See [11] for details of this approach. Alternate approaches to partial coherence include propagation of the cross spectral density (Fourier transform of Wigner function), macroparticle (MP) sampling of the electron beam distribution and coherent propagation, and Coherent Mode Decomposition(CMD), leading to a smaller number of coherent modes than the MP method. We will explore all these methods, comparing efficiency to decide on our partially coherent model. As a fast CMD method has recently been implented in SRW, we will be applying this to bending magnet and undulator beamlines and then performing the coherent propagation via LCT using the computed ABCD matrices from Shadow.

MACHINE LEARNING MODELS

Preliminary efforts to apply ML algorithms to learn a surrogate beamline model were reported in [1] in which a sample KB beamline was used to train a neural network with SRW simulations to reconstruct mirror errors. We have since generalized the tools for the SRW simulations, integrating them into our Sirepo GUI interface to SRW[3]. The data generated by the resulting simulations can then be used to train ML algorithms. We have included more elements in the

[2] `https://numpy.org/`
[3] `https://www.sirepo.com/srw#`

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV024

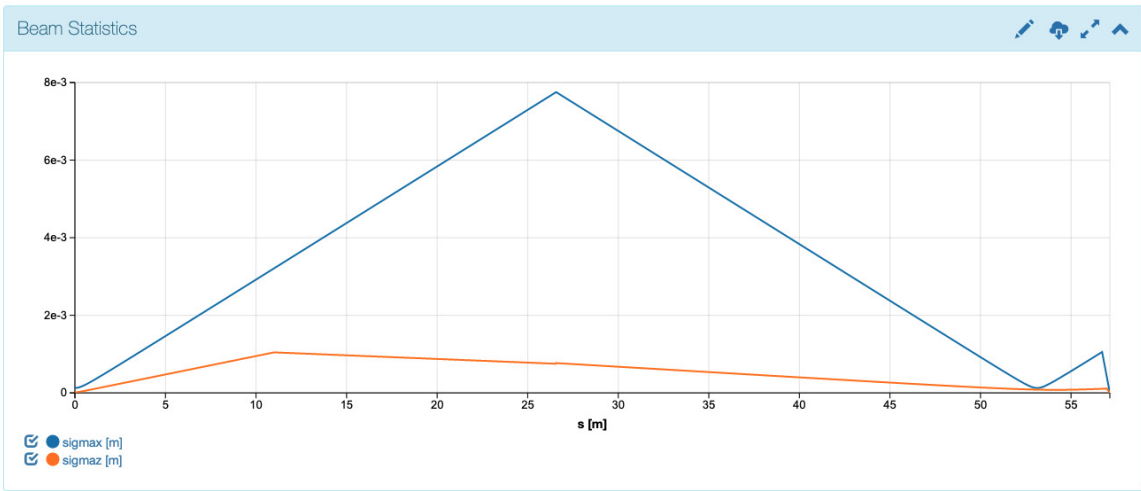


Figure 2: Horizontal (blue) and vertical (orange) beam-size propagation in the beam statistics report for the TES beamline.

interface for the data generating scripts initially developed as part of Phase I. In particular, we have implemented the following new elements and their corresponding parameters:

- aperture (position, and horizontal and vertical size);
- circular cylinder (position, rotation, and radius);
- toroidal mirror (position, rotation, and tangential and sagittal radii).

Figure 3 shows a screenshot of the interface for these new elements.

Continuing to develop our ML beamline capabilities, a subset of the NSLS-II development team worked on the Deep Beamline Simulation software.[4] That effort aims to train neural networks on solved problems, with the goal of lowering the computational cost of x-ray beamline simulations by using machine learning to simulate beamlines with a fidelity similar to SRW. We want to determine whether or not a neural network trained on data generated by previously created simulation tools (Sirepo-Bluesky, SRW, Shadow) can effectively identify relevant patterns and then solve newly-posed problems of similar type. In Fig. 4 we show some preliminary results from that effort. Tensorboard has been used to plot the loss function as the network learns the distribution resulting from varying a physical aperture in the beamline.

INTERFACE WITH BLUESKY BEAMLINE CONTROL SOFTWARE

The BlueSky software[5][12] allows beamline scientists to control their beamline while performing x-ray experiments. BlueSky plans are created which lead to motion of beamline components such as mirrors, monochromators, slits, etc. In order to connect the physical beamline to a simulated beamline, the Sirepo-Bluesky library.[6][13] has been created. Sirepo allows access to beamline simulation codes such as SRW and SHADOW, and will also include the `MABTrack` code when completed.

We have addressed a number of topics to improve the Sirepo-Bluesky library. When first developed, the library focused on supporting the SRW simulation code within Sirepo. After their recent refactoring of the library, the team added support for the Shadow simulation code within Sirepo. This effort has resulted in a more modular code: Support for SRW and Shadow are now implemented in separate Python modules, which provides for a cleaner separation between corresponding classes. This work included the creation of corresponding tests for the newly added features, as well as updates to the previous tests to ensure that new additions do not cause any regression or performance issues. As part of updating the testing framework, we changed the Continuous Integration (CI) service provider from TravisCI to GitHub Actions. This change allows us to run the tests free of charge, and also provides a more reliable and future-proof infrastructure.

Now that Shadow has been integrated into Sirepo-Bluesky, one may perform the same Bluesky scans as were formerly possible, but now with *either* SRW or Shadow as the underlying engine. Scans of interest include counts read from Sirepo's virtual "detector", step scans of a single parameter scanned over a range of values, and grid scans of multiple parameters over multiple ranges of values. In addition, the new integration of Shadow into Sirepo-Bluesky provides access to the Beam Statistics Report implemented in Sirepo-Shadow.

Because RadiaSoft personnel lack direct access to the BNL environment, but want to perform testing in as similar an environment as possible, we examined the possibility of using `sirepo.lib` (part of Sirepo) for working with Sirepo simulations in the absence of a server. Enhancements to the local development environment that better support Jupyterhub now allow developers to run Sirepo-Jupyter on their local machine for faster testing of new features in Sirepo-Bluesky.

[4] `https://github.com/NSLS-II/deep-beamline-simulation`
[5] `https://nsls-ii.github.io/bluesky/`
[6] `https://github.com/NSLS-II/sirepo-bluesky`

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV024

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

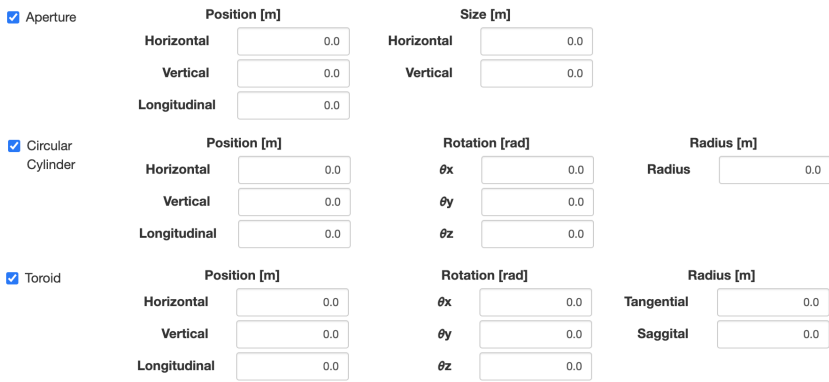


Figure 3: New beamline elements and their parameters for data generation. When the user provides a range of parameters, a script is produced that allows SRW to be run using `rsopt`, and will compute intensity data over the entire parameter range given.

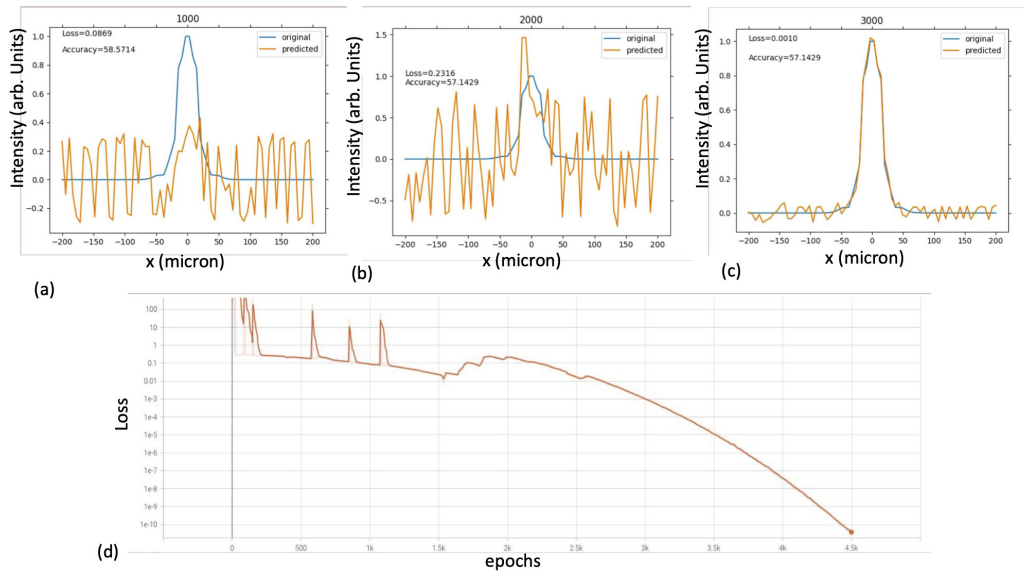


Figure 4: Tensorboard plots of loss function with varying epoch of (a)1000, (b)2000, and (c)3000. Gaussian radiation is produced in SRW, and a simple beamline with an aperture at 25 meters is varying, with the intensity observed at a watchpoint of 27 meters. (d) shows the accuracy as a function of all epochs during the learning process.

TES BEAMLINE MEASUREMENTS

Preliminary beam size measurements were carried out by Yonghua Du on the TES beamline, allowing us to begin comparison with the different types of propagation models. The TES beamline as set up in Sirepo SRW is show in figure citefig:TES.Sirepo.SRW. The layout for where the measurements were taken are shown in Figure 6. See [2] for more details about the layout and instrumentation of the TES beamline.

The beam size was measured at three positions: the fluorescent screen (FS) at ~ 50.242 m, the secondary slit aperture (SSA), and at the K-B mirror in the end station. The FS is mounted on the micrometer, which allows a measurement of the full horizontal beam size of ~ 1.5 mm. The beam size at the SSA was measured on the basis of SSA slit position and ion chamber (IC0) current. The full horizontal beam size is ~ 0.6 mm and vertical beam size is > 0.44 mm where ~ 50 % flux passed. The beam size at the K-B mirror was measured using the K-B step motor and the YAG crystal at the sample stage. The full horizontal beam size is ~ 1.1 mm and full vertical beam size is ~ 0.45 mm. During the measurement, we found that the vertical focusing point of the beam from toroidal mirror is between the KBV and sample position, not at the SSA. Comparison of these measurements to TES SRW simulations and moment propagation is ongoing, and improvements to the TES diagnostics are foreseen.

CONCLUSION

We have summarized our recent efforts to create reduced models suitable for an online model of x-ray beamlines and to allow access to these models integrated within the beamline control software, Bluesky. Working with the TES beam-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV024

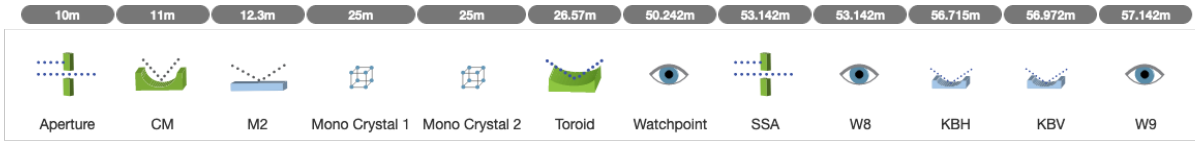


Figure 5: TES beamline in Sirepo SRW.

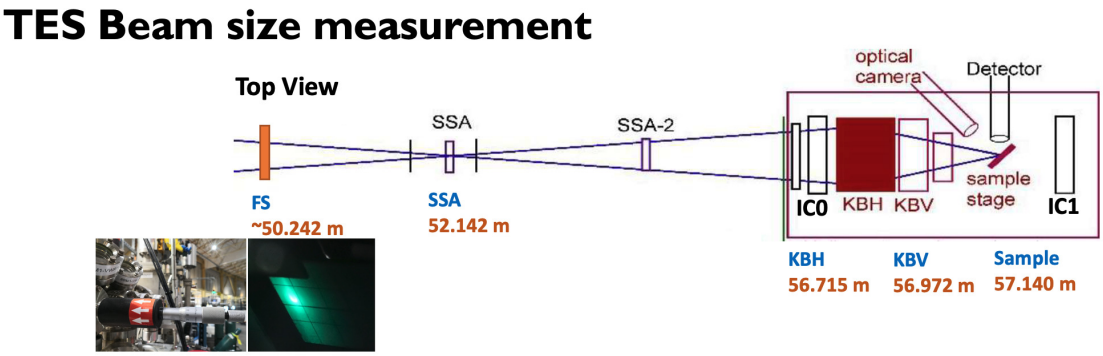


Figure 6: Layout for the TES beamsize measurements together with a photograph of the camera and resultant image on the fluorescent screen.

line, we are attempting a proof of principle demonstration of the creation of such an online model that will facilitate automated beamline control and reconfiguration.

ACKNOWLEDGMENTS

This work is supported, in part, by the US Department of Energy, Office of Science, Office of Nuclear Physics, including grant No. DE-SC0020593.

REFERENCES

[1] B. Nash, N. Goldring, J. Edelen, C. Federer, P. Moeller, and S. Webb, "Reduced model representation of x-ray transport suitable for beamline control," in *Advances in Computational Methods for X-Ray Optics V*, vol. 11493, International Society for Optics and Photonics, Aug. 2020, p. 114930C. DOI: 10.1117/12.2568187.

[2] P. Northrup, "The TES beamline (8-BM) at NSLS-II: Tender-energy spatially resolved X-ray absorption spectroscopy and X-ray fluorescence imaging," *J. Synchrotron Rad.*, vol. 26, no. 6, pp. 2064–2074, Nov. 2019. DOI: 10.1107/S1600577519012761.

[3] J. J. Healy, M. A. Kutay, H. M. Ozaktas, and J. T. Sheridan, Eds., *Linear Canonical Transforms: Theory and Applications*, ser. Springer Series in Optical Sciences. New York: Springer, 2016, vol. 198.

[4] B. M. Hennelly and J. T. Sheridan, "Fast numerical algorithm for the linear canonical transform," *J. Opt. Soc. Amer. A*, vol. 22, no. 5, pp. 928–937, 2005. DOI: 10.1364/JOSAA.22.000928.

[5] A. Koç, H. M. Ozaktas, C. Candan, and M. A. Kutay, "Digital computation of linear canonical transforms," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2383–2394, May 2008. DOI: 10.1109/TSP.2007.912890.

[6] A. Koç, H. M. Ozaktas, and L. Hesselink, "Fast and accurate algorithm for the computation of complex linear canonical transforms," *J. Opt. Soc. Amer. A*, vol. 27, no. 9, pp. 1896–1908, Sep. 2010. DOI: 10.1364/JOSAA.27.001896.

[7] J. J. Healy and J. T. Sheridan, "Fast linear canonical transforms," *J. Opt. Soc. Amer. A*, vol. 27, no. 1, pp. 21–30, Jan. 2010. DOI: 10.1364/JOSAA.27.000021.

[8] R. G. Campos and J. Figueroa, "A fast algorithm for the linear canonical transform," *Signal Process.*, vol. 91, no. 6, pp. 1444–1447, Jun. 2011. DOI: 10.1016/j.sigpro.2010.07.007.

[9] A. Koç, "Fast algorithms for digital computationof linear canonical transforms," PhD dissertation, Stanford University, Stanford, CA, Mar. 2011.

[10] S.-C. Pei and S.-G. Huang, "Fast discrete linear canonical transform based on CM-CC-CM decomposition and FFT," *IEEE Trans. Signal Process.*, vol. 64, no. 4, pp. 855–866, Feb. 2016. DOI: 10.1109/tsp.2015.2491891.

[11] B. Nash, N. Goldring, J. Edelen, S. Webb, and R. Celestre, "Propagation of partially coherent radiation using Wigner functions," *Phys. Rev. Accel. Beams*, vol. 24, no. 1, p. 010 702, Jan. 2021, ISSN: 2469-9888. DOI: 10.1103/PhysRevAccelBeams.24.010702.

[12] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's Ahead: A Multi-Facility Collaboration for an a la Carte Software Project for Data Acquisition and Management," *Synchrotron Radiation News*, vol. 32, no. 3, pp. 19–22, May 2019, ISSN: 0894-0886. DOI: 10.1080/08940886.2019.1608121.

[13] M. S. Rakitin *et al.*, "Introduction of the Sirepo-Bluesky interface and its application to the optimization problems," in *Advances in Computational Methods for X-Ray Optics V*, vol. 11493, International Society for Optics and Photonics, Aug. 2020, p. 1 149 311. DOI: 10.1117/12.2569000.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV025

INITIAL STUDIES OF CAVITY FAULT PREDICTION AT JEFFERSON LABORATORY*

L. S. Vidyaratne†, A. Carpenter, R. Suleiman, C. Tennant, D. Turner, Jefferson Laboratory, Newport News, VA, USA
K. Iftekharuddin, Md. Monibor Rahman, ODU Vision Lab, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA, USA

Abstract

The Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Laboratory is a CW recirculating linear accelerator (linac) that utilizes over 400 superconducting radio-frequency (SRF) cavities to accelerate electrons up to 12 GeV through 5-passes. Recent work has shown that given RF signals from a cavity during a fault as input, machine learning approaches can accurately classify the fault type. In this paper, we report initial results of predicting a fault onset using only data prior to the failure event. A dataset was constructed using time-series data immediately before a fault ("unstable") and 1.5 seconds prior to a fault ("stable") gathered from over 5,000 saved fault events. The data was used to train a binary classifier. The results gave key insights into the behaviour of several fault types and provided motivation to investigate whether data prior to a failure event could also predict the type of fault. We discuss our method using a sliding window approach. Based on encouraging initial results, we outline a path forward to leverage deep learning on streaming data for fault type prediction.

INTRODUCTION

The Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab is a high power, continuous wave recirculating linac servicing four different experimental nuclear physics end stations [1]. CEBAF completed an energy upgrade in 2017 with a goal of effectively doubling its energy reach from 6 GeV to 12 GeV. The upgrade required the installation of 11 additional cryomodules, named C100s denoting their capability for providing 100 MV of energy gain. A schematic of CEBAF with locations of the new C100 cryomodules is provided in Figure 1. Each cryomodule is composed of 8 superconducting radio-frequency (SRF) cavities. In addition, a digital low-level radio frequency system (LLRF) is implemented to regulate the new cryomodules.

CEBAF experiences frequent short machine downtime trips caused by SRF system faults, particularly when the cavity gradients are pushed to their upper limits. A significant portion of the SRF system faults occur within the C100 cryomodules. Consequently, a data acquisition system is implemented to record data from these cryomodules to investigate the nature and the origin of the SRF faults. The system is configured to record time-series waveform data when any of the C100 cavities fault. These recorded waveform data are analyzed by a subject matter expert (SME) to determine the cavity that caused the trip, and the type of fault. In previous studies we have investigated the possibility of using artificial intelligence (AI) techniques to automate this highly tedious waveform analysis process [2, 3]. It is quite helpful to expedite the beam recovery process with fast automated cavity fault identification after an event. However, it may be further beneficial to investigate the possibility of using AI to predict RF failures beforehand in order to reduce certain faults from occurring. While the data acquisition system is being upgraded for compatibility with such predictive models, we conduct a feasibility study for RF fault prediction using currently available data.

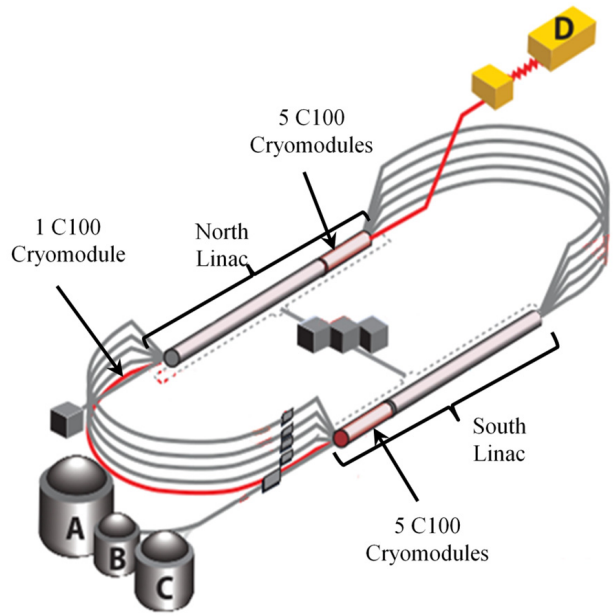


Figure 1: CEBAF schematic with locations of C100 cryomodules.

CAVITY FAULT CLASSIFICATION

The cavity fault classification is posed as a supervised machine learning (ML) problem, with ground truth fault labels for recorded data provided by SMEs. The data used for the classification task is the full time-series waveforms pertaining to each fault event recorded by the data acquisition system. The entire time-series waveform represents approximately 1638.4 ms (from t = –1536 ms to

* This work is supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Contract No. DE-AC05-06OR23177.
† lasithav@jlab.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV025

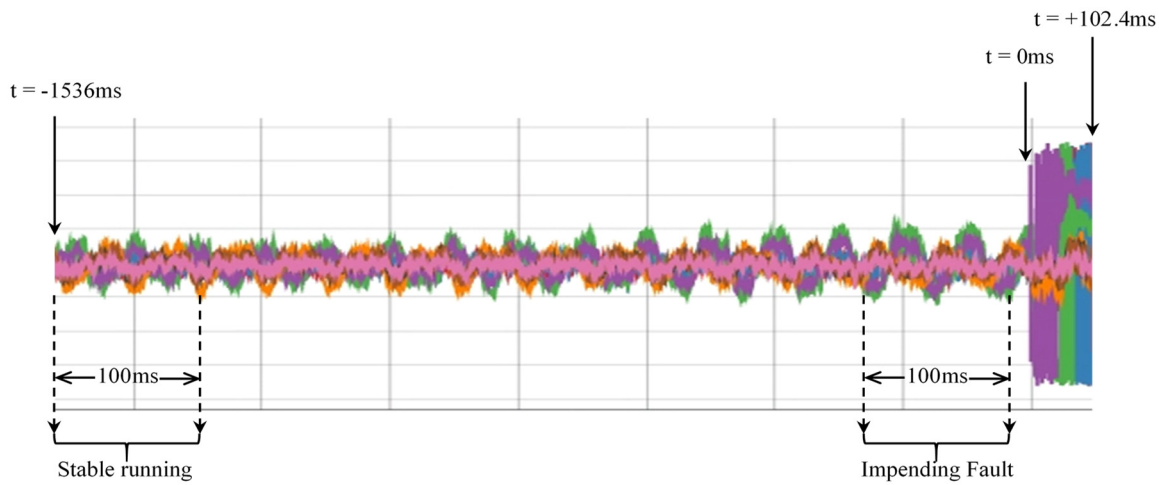


Figure 2: Waveform captured by the data acquisition system. The total duration of the waveform is 1.64 seconds.

t = +102.4 ms) comprised of both pre-fault and post-fault signals as shown in Figure 2. Waveforms are sampled at a constant rate of 5 kHz. We record 17 different RF signals from each C100 cavity, and a selection of these are used for analysis. In effect, the classification task uses both pre-fault and post-fault waveforms to identify the cavity that failed and the type of fault in a given cryomodule. Detailed discussions on the classical ML and deep learning (DL) models developed for cavity fault classification and associated performance characteristic can be found in [3, 4]. Table 1 summarizes the performance of ML and DL models for cavity fault classification with testing data. The best performing ML models are deployed within CEBAF.

Table 1: Cavity and Fault Type Classification Accuracy

	Cavity Identification	Fault Classification
ML Model (%)	88.0	86.9
DL Model (%)	87.8	81.3

Note that the classification task exploits both pre-fault and post-fault data to achieve the model accuracy shown in Table 1. However, a fault prediction model can only utilize data prior to a fault event to make predictions on an impending fault. Moreover, fault prediction must be performed at the cavity level as prediction of a fault would inherently identify the associated cavity. We therefore conduct the feasibility study for fault prediction as a two-step process as follows: 1) we define a binary classification task to identify waveforms describing impending faults, and 2) a moving window based multi-class classification task to predict the type of failure before onset.

BINARY CLASSIFICATION TASK

Binary classification task is designed to investigate the possibility of distinguishing waveforms representing imminent faults from waveforms representing stable running conditions.

Dataset

The dataset utilized for this study consists of a total of 5,047 fault events. Table 2 summarizes the dataset composition with respect to fault types.

Table 2: Fault Prediction Dataset

Fault Type	Number of Events
Single Cavity Turn Off	885
Microphonics	710
100 ms Quench	608
Controls Fault	847
Electronic Quench	673
3 ms Quench	542
Heat Riser Choke	720
Unknown	62

The current data acquisition system is set up to collect data only in the event of a RF cavity trip (see Figure 2). Note that approximately 94% of the captured waveform ($t < 0$) represents pre-fault activity, with $t = 0$ the fault onset, and $t > 0$ the post-fault data. Therefore, we utilize 100 ms segments extracted from the waveforms to represent stable running and impending fault classes as follows: for stable running, extract a 100 ms segment from the earliest possible window of the captured waveform (annotated as "Stable running" in Figure 2, with a window of [−1536 ms, −1436 ms]). Segments of this region are verified by SMEs as sufficient representations of stable running conditions. For the impending fault, we extract a 100 ms window as close as possible to the fault onset (annotated as "Impending fault" in Figure 2). Accordingly, we extract impending fault segments with a window of [−105 ms, −5 ms], with a 5 ms buffer to ensure that fault onset activity is not inadvertently captured. As a result, for the binary classification task we obtain 5,047 examples. Additionally, we retain labels of the underlying fault type in the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

waveforms representing the impending fault phase to perform a follow-up analysis on the feasibility of predicting fault types.

Prediction Model and Training

We develop a deep recurrent neural network (DRL) architecture to process the time-series for the binary classification task. The detailed architecture is shown in Figure 3.

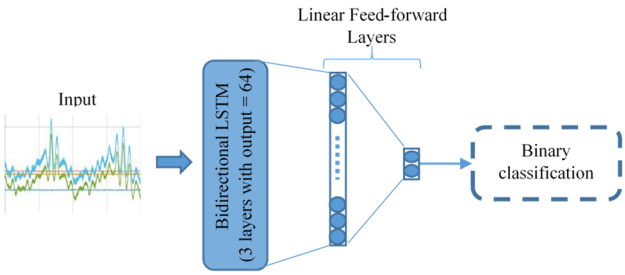


Figure 3: DRL architecture for binary classification of pre-fault data.

The 100 ms segments extracted for this study represents time-series data with each 500 time steps long. The Long Short Term Memory (LSTM) [5] layers in the above model processes the time-series input, and the corresponding features are classified at the last layer (softmax activation). The architecture and the training scheme is developed using the PyTorch deep learning library [6]. The data is divided with fault based stratification to obtain 60% for training, 20% for validation, and the remaining 20% for testing.

Binary Classification Results

The confusion matrix obtained from the testing data is shown in Figure 4. The overall testing data classification accuracy is 74.74%. Detailed testing results are summarized in Table 3. However, the confusion matrix shows a distinct issue in the models ability to recognize many impending fault events correctly, highlighted by the large false negative percentage (23.48%). This is also observed in Table 3 where the recall rate for impending fault, and the precision for stable running classes are both low.

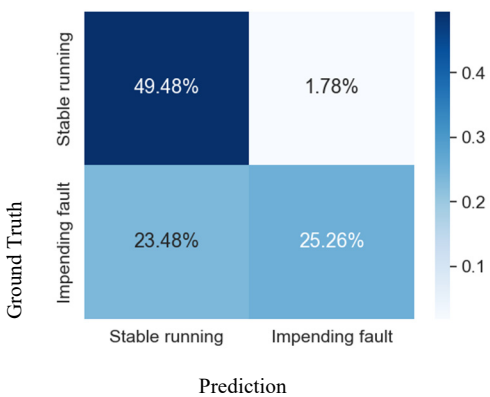


Figure 4: Confusion matrix for classification of stable running versus impending fault.

Table 3: Binary Classification Performance

Class	Precision	Recall	F1-score
Stable running	67.8%	96.5%	79.6%
Impending fault	93.4%	51.8%	66.7%

This behavior indicates that certain impending fault segments may closely resemble stable running conditions prior to fault onset, to a degree that the model is unable to distinguish between the two classes. We speculate that certain fault types identified by SMEs may not present sufficient precursors within the waveforms we use for the analysis. In order to further investigate this issue, we take a closer look at the fault types that represent the false negatives. Figure 5 shows the distribution of false negative events in terms of the ground truth fault type.

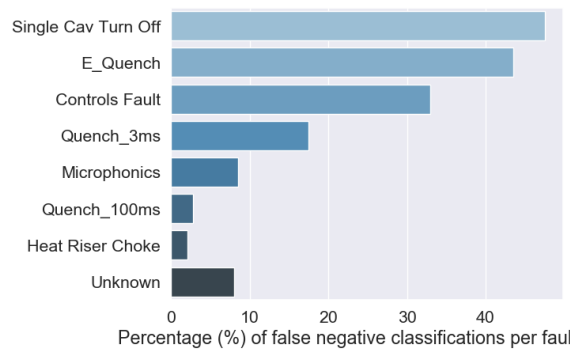


Figure 5: Percentage of events with false negative classification according to fault type.

It is evident from Figure 5 that the false negatives are dominated by events from 'Single Cav Turn Off', 'E_Quench', and 'Controls Fault' fault types. Upon discussion with domain experts, it was confirmed that these fault types oftentimes require ancillary data for correct identification due to the lack of precursors present in the waveforms used for the study. Therefore we conducted a secondary experiment where we discard events from the three aforementioned fault types and obtain a secondary dataset of 2,642 fault events. We use the same model architecture, and follow the same training procedure. The model achieves a classification accuracy 92.1% for this dataset.

FAULT TYPE PREDICTION TASK

Given encouraging results from the binary classification task, we investigate the possibility of predicting the fault type before the onset of beam trip. In order to quantify the ability to predict the fault type with respect to the onset, we define a window-based analysis on the waveform data that we have collected. Specifically, a window of a specific

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV025

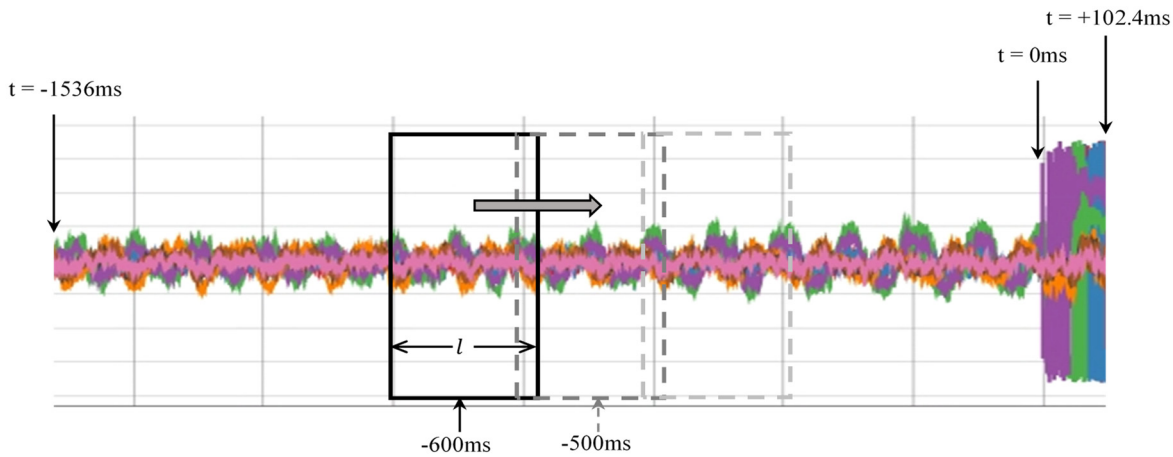


Figure 6: Window based fault prediction analysis scheme.

length l is positioned over the waveform data, centered on a time stamp t before the onset of the fault, as illustrated in Figure 6. The data that falls within the window is extracted to perform analysis on fault prediction. This is repeated over window locations that incrementally move from t = −600 ms to 0 ms in 50 ms intervals. The value of −600 ms is chosen as the furthest time stamp prior to the fault event based on suggestions of domain experts, and 0 ms denotes the fault event onset.

Dataset

We utilize the same dataset described in Table 2 as our base dataset. The window-based analysis in Figure 6 results in multiple experiments conducted over each window location. In essence, for a given window location t, data that falls within the window is extracted from all examples to create training, validation, and testing sets specific to location t. Analysis using this data yields fault prediction performance for t ms prior to fault onset. In addition to multiple window locations, we also experiment with the effect of window size on prediction performance using lengths of l = 100 ms, 200 ms, 300 ms.

Prediction Model and Training

We utilize the same general deep LSTM architecture shown in Figure 3 for this analysis. However, fault type prediction is essentially a multiclass classification task. Therefore, we augment the classification layer of the network to accommodate 8 fault classes. The same training and testing procedures used in the binary classification task are leveraged for each window dataset. Model development, training, and testing is performed using the PyTorch library [6].

Fault Type Prediction Results

Although the prediction network is trained and tested for each window experiment in a multiclass manner, we inspect the prediction performance for each class separately. This allows for observing the prediction efficacy for each fault more efficiently, considering the insights obtained through the binary classification task. Figure 7 shows performance of the prediction model plotted as a function of time prior to onset of "Microphonics", and "E_Quench" fault types. The horizontal axis of the plots shows the center of the window used to extract data for the prediction task. For instance,

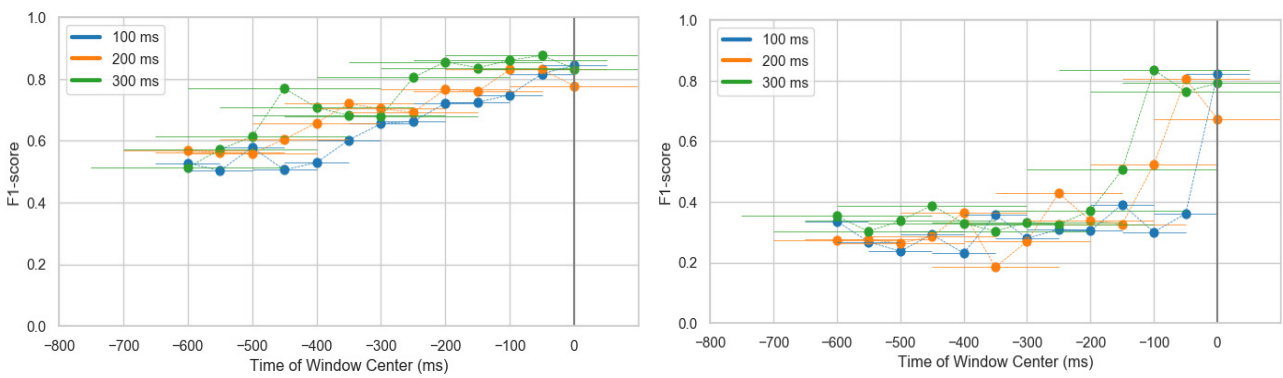


Figure 7: F1-score plots for the window based fault type prediction of microphonics and electronic quench fault types.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the F1-score at –600 ms for a given fault type indicates the model's ability to predict the fault type 600 ms prior to the fault onset. We also observe that the predictive power of the model is highly dependent of the type of fault. For instance, the model is able to predict microphonics faults

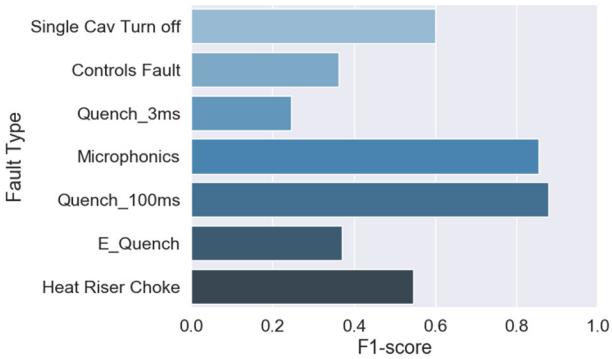


Figure 8: Fault specific model prediction performance for the case of 300 ms data window centered at –200 ms.

with over 0.5 F1-score from –600 ms with significant improvements with window locations closer to onset. Conversely, the model is unable to predict E_Quench events with acceptable accuracy until the window overlaps the fault onset. Figure 8 shows a snapshot of the fault dependent prediction performance for the case of 300 ms data window centered at –200 ms (200 ms before the fault onset).

This analysis further demonstrates that certain faults such as microphonics show precursors prior to onset, which can be used to predict the faults prior to a trip. The analysis also provides insights into several fault types that may not show precursors due to their nature, or may not be sufficiently captured in the waveforms used in this study.

FUTURE WORK

We have discussed our initial studies into the possibility of automated prediction of RF faults in C100 cavities using

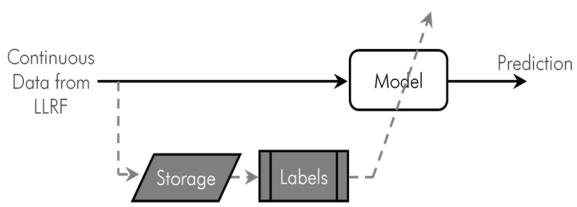


Figure 9: Preliminary framework for using machine learning models with continuously streaming data.

ML models. The investigations have yielded promising results, with insights into the possible prediction time ranges for multiple fault types. However, this study leverages static RF fault data. An effective fault prediction framework would require a data acquisition system that is able to stream the data signals in real-time during CEBAF operation. The C100 modules, and the associated LLRF systems, are currently undergoing a firmware upgrade to allow the collection of streaming data. This will provide valuable information regarding variations encountered in stable running conditions, and possible non-RF related fault events in CEBAF.

Streaming data provides an opportunity to further improve and adapt the machine learning models to predict faults before they occur. To that end, we envision a data processing framework as shown in Figure 9. According to the preliminary framework, the prediction model gets direct access to the continuous data to make predictions on stable running versus impending fault, with probable fault type in real-time. The framework also allows storing select subset of the data for the purposes of routine model performance evaluation, developing new models, re-training and fine-tuning of the deployed models. We also envision the C100 firmware upgrades and the proposed streaming data processing framework will generate significant amounts of information-rich data that may be useful to diagnose other CEBAF machine events beyond SRF cavity faults.

REFERENCES

[1] C. E. Reece, "Continuous wave superconducting radio frequency electron linac for nuclear physics research," *Physical Review Accelerators and Beams,* vol. 19, no. 12, p. 124801, 12/28/ 2016, doi: 10.1103/PhysRevAccelBeams.19.124801

[2] A. Solopova *et al.*, "SRF cavity fault classification using machine learning at CEBAF," in *10th Int. Particle Accelerator Conf.(IPAC'19), Melbourne, Australia, 19-24 May 2019*, 2019: JACOW Publishing, Geneva, Switzerland, pp. 1167-1170, doi:10.18429/JACoW-IPAC2019-TUXXPLM2

[3] C. Tennant, A. Carpenter, T. Powers, A. Shabalina Solopova, L. Vidyaratne, and K. Iftekharuddin, "Superconducting radio-frequency cavity fault classification using machine learning at Jefferson Laboratory," *Physical Review Accelerators and Beams,* vol. 23, no. 11, p. 114601, 11/30/ 2020, doi: 10.1103/PhysRevAccelBeams.23.114601

[4] L. Vidyaratne, T. Powers, C. Tennant, K. M. Iftekharuddin, Md M. Rahman and A. Shabalina, "Deep Learning-Based Superconducting Radio-Frequency Cavity Fault Identification at Jefferson Laboratory," *Frontiers Operational Intelligence (under review),* 2021.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[6] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems,* vol. 32, pp. 8026-8037, 2019.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV026

MULTI-CHANNEL HEATERS DRIVER FOR SIRIUS BEAMLINE'S OPTICAL DEVICES

M. M. Donatti†, D. H. C. Araujo, F. H. Cardoso, G. B. Z. L. Moreno, L. Sanfelici, G. T. Semissatto,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

Thermal management of optomechanical devices, such as mirrors and monochromators, is one of the main bottlenecks in the overall performance of many X-Rays beamlines, particularly for Sirius: the new 4th generation Brazilian synchrotron light source [1]. Due to high intensity photon beams some optical devices need to be cryogenically cooled and a closed-loop temperature control must be implemented to reduce mechanical distortions and instabilities. This work aims to describe the hardware design of a multi-channel driver for vacuum-ready ohmic heaters used in critical optical elements. The device receives PWM (Pulse-Width Modulation) signals and can control up to 8 heaters individually. Interlocks and failure management were implemented using digital signals input/outputs available at device rear panel. The driver is equipped with a software programmable current limiter to prevent load overheating and it has voltage/current diagnostics monitored via EPICS (Experimental Physics and Industrial Control System) or an embedded HTTP (Hypertext Transfer Protocol) server. Enclosed in a 1U (rack unit) rack mount case, the driver can deliver up to 2A per channel in 12V and 24V output voltage versions. Performance measurements will be presented to evaluate functionalities, noise, linearity and bandwidth response.

INTRODUCTION

The optical devices design for synchrotron facilities is clear to be one of the most discussed topics to ensure state-of-art beamlines and perform complex experiments, especially for Sirius, the new low-emittance 4th generation Brazilian synchrotron light source [1]. These devices are highly stable thermo-mechanical solutions equipped with cryocooling schemes via cryostats and heaters for temperature control. This work aims to describe the hardware design of a multi-channel heaters drivers for Sirius custom high-performance mirrors [2] and monochromators [3-4]. The driver is also allocated at experimental stations with cryogenic setups, such as Tarumã [5], a sub-microprobe station at CARNAUBA (Coherent X-Ray Nanoprobe Beamline) beamline.

The multichannel driver was designed to simultaneously deliver up to 1.5A at 12V to eight ohmic vacuum-compatible heaters. Each channel can be individually controlled by a TTL (Transistor-Transistor Logic) compatible PWM (Pulse-Width Modulation) signal. Optocoupled interlock inputs were implemented for safety purposes and failure outputs for open load and short-circuit signaling. A current limiter scheme has been included to protect both heater and driver from overload and short-circuits.

Figure 1 shows the internal block diagram and interfaces. Voltage, current and duty cycle are monitored by two multi-channel 12-bit ADC (Analog-to-Digital Converter) AMC7812B [6], also responsible to set current limiter threshold over an internal DAC (Digital-to-Analog Converter). The AMC7812B are connected via SPI (Serial Peripheral Interface) to a NXP LPC1768 [7], an ARM® Cortex-M3 microcontroller.

Ethernet connection is provided to remotely access diagnostics data via an embedded HTTP (Hypertext Transfer Protocol) webserver and a TCP (Transmission Control Protocol) socket for EPICS integration. The current limiter threshold and a software enable control are available remotely. An USB (Universal Serial Bus) connector is provided at front panel for firmware update purpose.

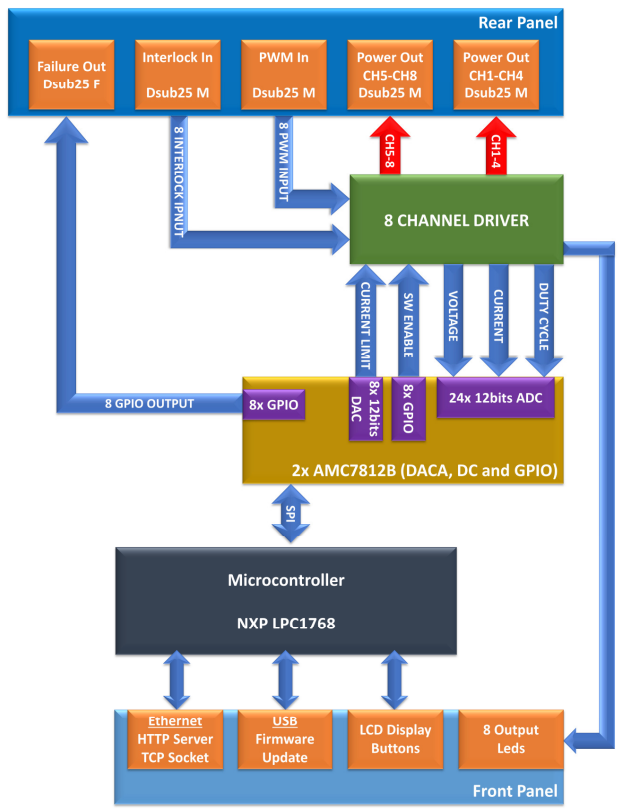


Figure 1: 8-Channel Heaters Driver Block Diagram.

This paper is organized as follows: Section II deals with hardware design details and concepts; Section III presents the embedded firmware architecture; Section IV discuss the mechanical design; Section V presents general results including performance parameters and Sirius'

† mauricio.donatti@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV026

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

monochromator case study. Finally, Section V outlines the main conclusions of this work.

HARDWARE DESIGN

Power Driver Topology

The power driver is based on a simple buck DC-DC voltage regulator [8], which has its output voltage controlled by an input voltage (V_C) applied to a series resistor (R_C) at the feedback node, as shown in Fig. 2:

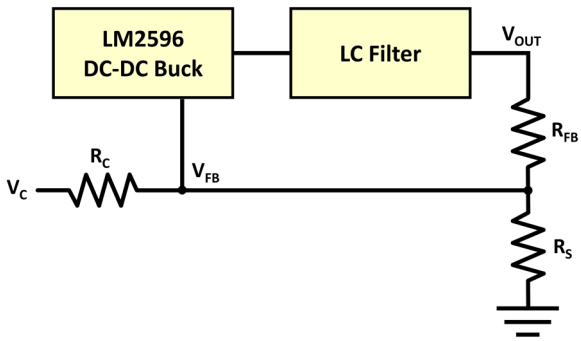


Figure 2: LM2596 Control Principle.

Assuming a DC-DC stable control loop, the feedback node voltage (V_{FB}) is fixed at 1.3V [8]. Considering the current drawn by the regulator feedback node (I_{FB} 10nA typical [8]) can be neglected, the output voltage is a linear function of control voltage (V_C) described by Eq. (1):

$$\frac{V_{OUT}-V_{FB}}{R_{FB}} = \frac{V_{FB}}{R_S} + \frac{V_{FB}-V_C}{R_C} \quad (1)$$

The maximum output voltage can be obtained by Eq. (2), assuming V_C at 0V and LM2596 input voltage higher than V_{OUT_MAX}:

$$V_{OUT_MAX} = V_{FB}\left(1 + \frac{R_{FB}(R_S+R_C)}{R_S R_C}\right) \quad (2)$$

Similarly, assuming 0V the desired minimum output voltage, solving Eq. (1) at this condition, the maximum control voltage can be obtained, as shown in Eq. 3:

$$V_{C_MAX} = V_{FB}\left(1 + \frac{R_C(R_S+R_{FB})}{R_S R_{FB}}\right) \quad (3)$$

These equations were used to determine resistors values to 12V maximum output voltage at 15V power supply. The control voltage is generated by a PWM signal applied to a Sallen-Key second-order low-pass active filter [9] (C_1=C_2=100nF and R_1=R_2=100kΩ) designed to limit bandwidth at 16Hz, transfer function shown by Eq. (4):

$$H(s) = \frac{1}{1+2RCs+(RCs)^2} = \frac{1}{1+0.02s+10^{-4}s^2} \quad (4)$$

The control bandwidth is expected to be limited by this filter, once the internal control loop response of the bulk regulator is considerably faster than 16Hz.

A current limiter is designed based on an analog comparator switching a PNP transistor that injects current directly at LM2596 feedback node. Since the load current overcomes the configured DAC threshold, the transistor turns on, dropping the LM2596 output voltage.

The power driver has been designed to provide 12V output voltage, however the hardware can be easily adapted to provide 24V output. This modification allows each channel to deliver up to 36W (1.5A at 24V) for higher power heaters, as used in Tarumã cryogenic experimental station [5].

Signal Acquisition

Multiple channel 12-bit ADCs (AMC7812B) acquire 24 analog voltages. Conditioning circuits with rail-to-rail single supply operational amplifiers were designed for ADC protection and signal conditioning. Therefore, the device perform 3 measurements per channel:

- The voltage across a shunt resistor, measuring the output current indirectly.
- The output voltage.
- The control voltage (Fig. 2 V_C), inversely proportional to the PWM input duty cycle.

Passive anti-aliasing filters are included at ADC inputs.

Additional Features

Optocoupled normally open enable inputs were designed to prevent power deliver in critical failure situations. These inputs are tied directly to LM2596 enable pins, that is, they are not firmware dependents.

Based on voltage and current diagnostics is possible to deduce open-load and short-circuit failures, triggering 5V TTL compatible optocoupled outputs at the rear panel.

FIRMWARE ARCHITECTURE

The microcontroller LPC1768 is programmed using Mbed framework [10], running Mbed Os 2 as real-time operating system, arranging tasks in 4 separate threads:

- **Main thread**: running at normal priority, responsible for Ethernet reconnection in case of failure, device startup configuration and save persistent data into internal Flash.
- **Sampling thread**: running at real-time priority, this thread samples ADCs at 100 samples per second and applies a digital filter.
- **Webserver socket thread**: running at normal priority, a TCP socket server is available at port 80, in which a HTTP server is implemented for quick diagnostics.
- **TCP socket thread**: running at normal priority, this thread implements a TCP socket server at port 6767 with a question/answer protocol command list to configure the device and read diagnostics data. The IOC (Input/Output Controller) makes use of this connection to integrate the device with EPICS control system.

The sampling thread performs an exponential moving average IIR (Infinite Impulse Response) filter, as shown in Eq. (5).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV026

$$y[n] = (1-\alpha)y[n-1] + \alpha x[n] \quad (5)$$

Considering 100 samples per second and α equals to 0.17, the cutoff frequency is 3Hz, as shown by Eq. (6):

$$f_{3dB} = \frac{F_S}{2\pi}\cos^{-1}\left(1 - \frac{\alpha^2}{2(1-\alpha)}\right) \cong 3Hz \quad (6)$$

MECHANICAL DESIGN

The driver is enclosed in a 1U 19" rack mount metallic case, as shown in Fig. 3. Two power supplies provide board voltages:

- 15V up to 13.4A (200W) to power drivers.
- 5V up to 5A (25W) to logic and I/O's (Inputs and Outputs).

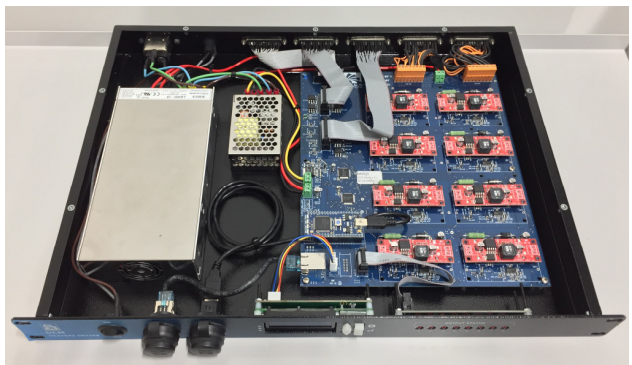


Figure 3: Mechanical Assembly.

The front panel, shown in Fig. 4, displays channels status, provides a main switch, ethernet input and USB input for firmware updates.

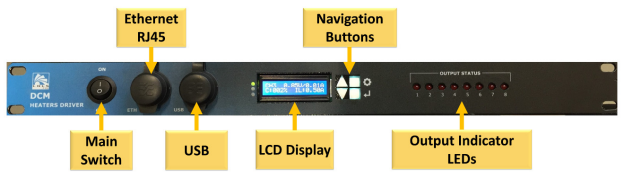


Figure 4: Front Panel.

At the rear panel, shown in Fig. 5, the AC line input and 2A fuse are connected to power supplies and 5 25-pin Dsub connectors provide output power to heaters and I/O's to controller: Control Input, Failure Output, Interlock/Enable Input, Power Output (2x).

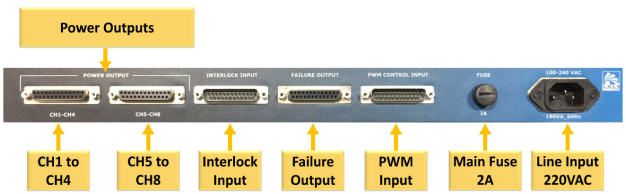


Figure 5: Rear Panel.

RESULTS

Electronics characterization

Due to hardware limitations, the output response is linear from 10% to 85% PWM duty cycle, as shown in Fig. 6, obtained delivering power to 20Ω loads. PWM frequency has been adjusted to 1kHz.

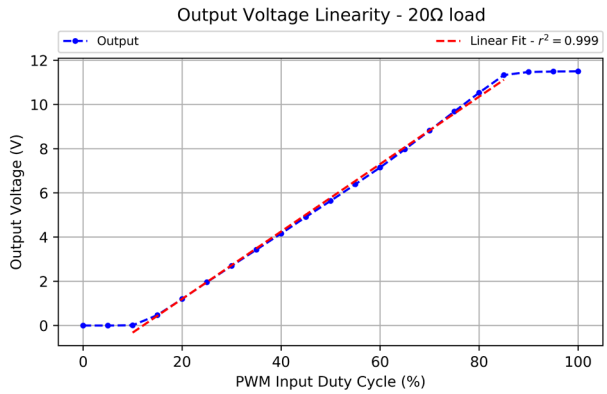


Figure 6: Output Linearity.

As mentioned in Hardware Design section, the driver bandwidth is limited by the Sallen-Key filter, designed to cut-off over 16Hz. Figure 7 shows driver frequency response, that has been obtained using a sinusoidal modulated 1kHz PWM signal delivering power to 20Ω loads. The sinusoidal signal modulates from 20% to 80% duty cycle with 100mHz to 100Hz variation. The normalized output refers to peak-to-peak AC load voltage at a given frequency.

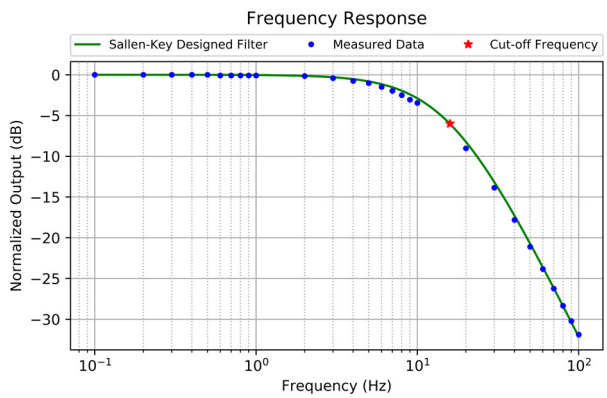


Figure 7: Single-channel Frequency Response.

The null-input output noise presents 100kHz 60mVpp spikes, shown in Fig. 8. At this condition, 31.75mV DC voltage is applied to 20Ω loads.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV026

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

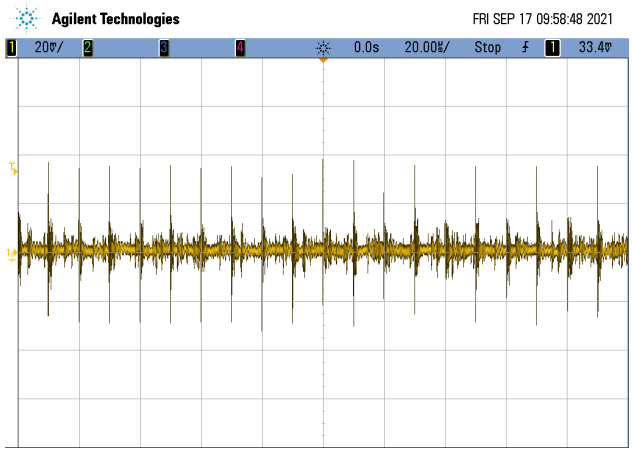


Figure 8: Output Voltage (Shorted-input and 20Ω load, AC coupling).

The output noise worst case occurs at 50% duty cycle input, 200mVpp spikes at 1kHz PWM frequency are shown in Fig. 9. In this condition, the driver delivers 5.89V to a 20Ω load.

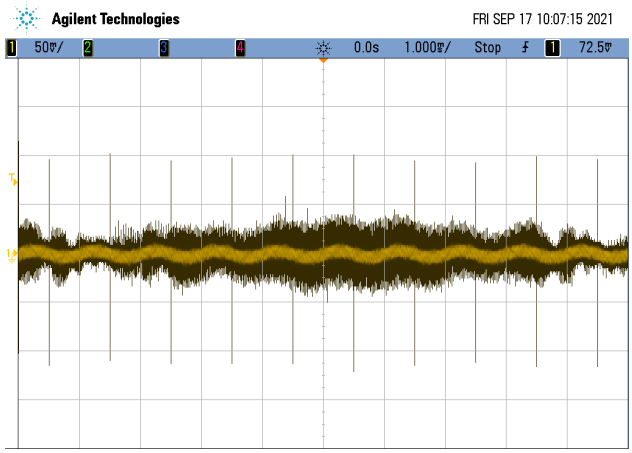


Figure 9: Output Voltage (1kHz 50% PWM Input delivering 5.89V to 20Ω load, AC coupling).

Temperature variations involved in thermal systems are usually long-term changes, thus tens of mV spikes at kHz frequencies does not affect overall performance.

The enable input is necessary to avoid unexpected overtemperatures in case of controller's failure. The output rise and fall times have been obtained by adjusting 100% duty cycle PWM inputs and applying a 5Hz square wave to enable inputs. Results are shown in Table 1.

Table 1: Enable Input Timing Performance

Enable	95% Settling Time
Rise edge	44ms
Fall edge	51ms

Field Application Results

The heaters driver is employed in several optical devices at Sirius beamlines. The simplified block diagram of the Double Crystal Monochromator (HD-DCM) [3] thermal management control system is presented in Fig. 10, using a NI CompactRIO [11] as main controller:

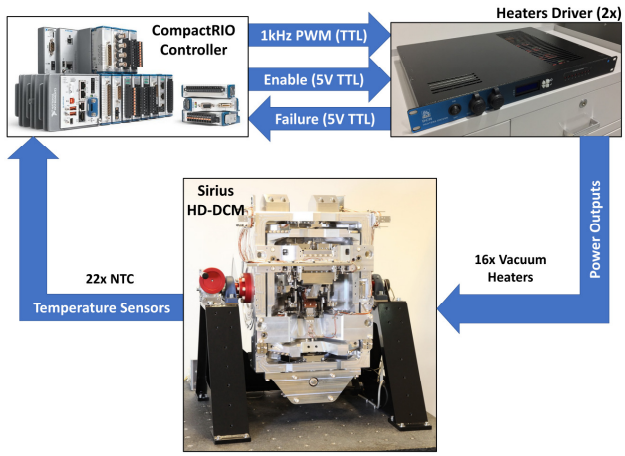


Figure 10: Sirius HD-DCM heaters driver application simplified block diagram.

A closed loop 24-hour long-term thermal stability is shown in Fig. 11, spotting DCM crystals temperature sensors during a regular user shift at EMA (Extreme condition Methods of Analysis) beamline [12]:

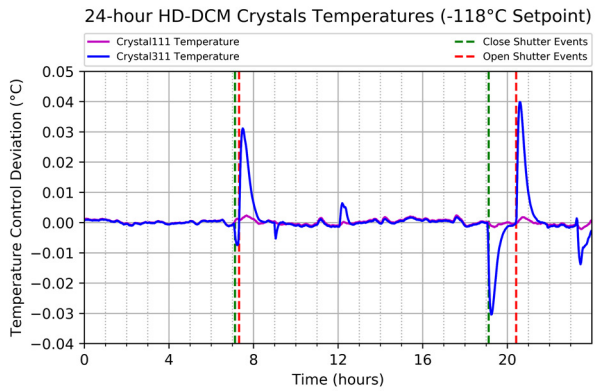


Figure 11: 24-hour HD-DCM crystal temperature measurements.

There is a strong correlation between variations and gamma shutter events, nevertheless crystals temperatures remain below 0.05ºC deviation over 24 hours.

The HD-DCM thermal system characterization and control loop tuning are not intended to be presented at this paper, however related works [4, 13] discuss this subject in-depth.

CONCLUSION

This work presented the hardware design concepts and results about a multi-channel heaters driver, intended to be used at beamlines optical devices, such as mirrors and monochromators. The driver is capable to deliver up to 18W per channel at 12V (36W per channel at 24V) simultaneously to eight ohmic heaters. The output power is controlled by 5V PWM signals that can be easily integrated to any digital controller.

In addition, 5V TTL failure outputs signalize malfunction heaters and enable inputs turn off power outputs 51ms afterwards a falling edge. An embedded webserver is

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV026

implemented to allow fast diagnostics and a TCP socket is available for EPICS infrastructure integration.

Power outputs noise levels have not significant influence for thermal systems, typically characterized with long-term variations.

The first 20 units production set currently being used in first 6 Sirius` beamlines were manufactured in 2018. In 2021, 12 additional units have been produced for upcoming beamlines. Driver hardware failures have not been reported in the past years, supporting driver high reliability, usually an essential requirement for thermal management systems.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the Brazilian Ministry of Science and Technology as funding agency and all the engineering and science teams members working for the Sirius project.

REFERENCES

[1] L. Liu, X. R. Resende, and F. H. de Sá, "A New Optics for Sirius", in *Proc. 7th Int. Particle Accelerator Conf. (IPAC'16)*, Busan, Korea, May 2016, pp. 3413-3416. doi:10.18429/JACoW-IPAC2016-THPMR013

[2] R. R. Geraldes *et al.*, "The design of exactly-constrained x-ray mirror systems for Sirius" in *Proc. 10th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'18)*, Paris, France, Jun 2018. pp 173-178. doi:10.18429/JACoW-MEDSI2018-WEOAMA04

[3] R. R. Geraldes *et al.*, "The New High Dynamics DCM for Sirius", in *Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'16)*, Barcelona, Spain, Sep. 2016, pp. 141-146. doi:10.18429/JACoW-MEDSI2016-TUCA05

[4] M. Saveri Silva, R. R. Geraldes, A. Gilmour, T. A. M. Ruijl, and R. M. Schneider, "Thermal Management and Crystal Clamping Concepts for the New High-Dynamics DCM for Sirius", in *Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'16)*, Barcelona, Spain, Sep. 2016, pp. 194-197. doi:10.18429/JACoW-MEDSI2016-TUPE15

[5] F. R. Lena *et al.*, "A Cryogenic Sample Environment for the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at the 11th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Int. Conf. (MEDSI'20), Chicago, USA, Jul. 2021, paper WEPC02.

[6] Texas Instruments, AMC7812B 12-Bit Analog Monitoring and Control Solution with Multichannel ADC, DACs, and Temperature Sensors, https://www.ti.com/lit/ds/symlink/amc7812b.pdf

[7] NXP, Arm LPC1768 Board, https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc1700-cortex-m3/arm-mbed-lpc1768-board:OM11043

[8] Texas Instruments, LM2596 SIMPLE SWITCHER Power Converter 150kHz, 3A Step-Down Voltage Regulator, https://www.ti.com/lit/ds/symlink/lm2596.pdf

[9] Texas Instruments, Application Report: Analysis of the Sallen-Key Architecture, Sep. 2002, https://www.ti.com/lit/an/sloa024b/sloa024b.pdf

[10] MBED platform, https://os.mbed.com/

[11] National Instruments, CompactRIO Platform Overview, https://www.ni.com/pt-br/shop/compactrio.html

[12] R. D. dos Reis *et al.*, "Preliminary Overview of the Extreme Condition Beamline (EMA) at the new Brazilian Synchrotron Source (Sirius)", in *Journal of Physics: Conference Series 2020*, Rio de Janeiro, Brazil, Aug. 2019, Vol 1609 012015. doi: 10.1088/1742-6596/1609/1/012015

[13] J. L. N. Brito *et al.*, "Temperature Control for Precision Beamline Systems of Sirius/LNLS", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, October 2021, paper WEPV001, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV027

EXPANDABLE AND MODULAR MONITORING AND ACTUATION SYSTEM FOR ENGINEERING CABINETS AT SIRIUS LIGHT SOURCE

P. H. Nallin†, J. G. R. S. Franco, G. F. Freitas, R. W. Polli
Brazilian Center for Research in Energy and Materials, Campinas, Brazil

Abstract

Having multipurpose hardware architectures for controls and monitoring systems has become a need nowadays. When it comes to modular and easily expandable devices, it brings together a system which is easy to maintain and can reach many applications. Concerning Sirius accelerators, which is a 4th generation light source, monitoring environment variables becomes crucial when it comes to accelerator stability and reliability. Several cabinets make up engineering infrastructure, therefore, monitoring, and acting over their environment such as internal temperature, pressure, and fan status, increases overall system reliability. This paper presents an inexpensive hardware topology to deal with multiple sensors and actuators mainly designed to monitor cabinets and prevent beam quality losses due to equipment faults.

INTRODUCTION

Sirius accelerator, the new 4th generation Brazilian synchrotron light source, has been under commissioning since 2019. Some improvements and maintenance have been accomplished for increasing machine performance and reliability in various subsystems.

Although the machine has not reached its project parameters yet, internal, and external scientists can run their experiments in available beamlines infrastructure, along with commissioning and maintenance shifts.

Both commissioning and user shifts require subsystems to work properly. For this purpose, it is necessary to have more information about the environment where equipment is located. The more scalable andeasier to expand the solution is, the more useful it will be regarding future needs. The concept of a multipurpose monitoring and actuating system, which will be initially installed in the Engineering Area, leads to acquiring a large variety of signals, making it easy to prevent, detect or even predict misfunction of a specific device.

OVERVIEW AND MOTIVATIONS

In order tohave a bright and stable photon bean, it is also necessary to keep the accelerators' installation area stable and reliable, which leads to the monitoring of some environment variables, such as the temperature of many different locations, the humidity where each equipment is located, the ambient pressure and several other data.

Sirius' monitoring network, an in-house project called SIMAR (Cabinet Monitoring and Actuation System), has been under development and its main purpose is to obtain data information from the cabinets, which store a large variety of equipment used in Sirius subsystems, such as vacuum, diagnostics and power supply equipment.

SYSTEM ARCHITECTURE

SIMAR is a multipurpose project, based on a Beagle Board single board computer, widely used in Sirius Control System [1]. It aims to handle multiple modular hardware, each one designed to run independently, making it possible to use various sensors and actuatorsin the same system and, also, flexible enough to operate in different conditions. The figure 1 shows the structure of the SIMAR project.

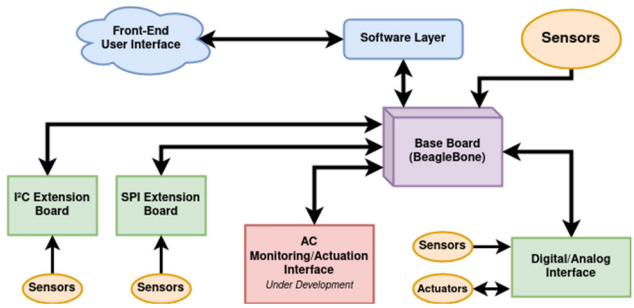


Figure 1: SIMAR conceptual design. It can integrate multiple devices, extend digital communication and easily scalable.

According to the environment to be monitored, additional stacked boards may be added as well as addressable splitters for multidrop digital communication channels. Every board must be hardware-configured prior to any new installation, in order to have a unique address identification.

Considering BeagleBone software integration, it is quite satisfactory to have one single disk image for all nodes in Controls Network, based on Debian. For that purpose, SIMAR has also been integrated into the BeagleBone-functionality-discovery service, which runs at initialization and gets information about hardware and equipment connected to a node. Then, all necessary applications are run.

HARDWARE DEVELOPMENTS

Base Board

The base board is the SIMAR main stack hardware project. It is responsible for functioning as a master on the communication buses and input/output pins. It also integrates the Programmable Real-time Units (PRUs) [2] into SIMAR interface for real-time data processing.

†patricia.nallin@cnpem.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV027

Four serial protocol signals (SPI, I²C, UART and 1-wire) are available on the base stack and they can be used for communication with several devices and sensors.

Digital and Analog Interface Stack

The first auxiliary board for SIMAR project is a general-purpose input/output digital and input analog data interface. This board makes it possible to write and read an 8-bit digital word and read up to six analog signal channels. This interface stack is controlled by the base board, as shown in figure 2, through SPI communication. Among the transmitted data, one byte is used to select the interface board, according to its unique address and to select the functionality inside of the interface board (for example, read or write digital data). If necessary, one byte for updating the digital outputs.

To write or read digital data, SPI registers (serial-in/parallel-out and parallel-in/serial-out) are available on the board.

For reading analog signals, the BeagleBone's internal 12-bit ADCs are used. Since the original input range is from 0V up to 1.8 V, signal conditioning and protecting circuits are included in order to increase voltage range and, also, prevent over voltage on BeagleBone pins.

In addition to communicating with external signals, this stack has an integrated SPI EEPROM memory, making it possible to store the latest output values to be recovered after a power outage, general board information and further necessities. The interface board is compatible with many SPI EEPROM models.

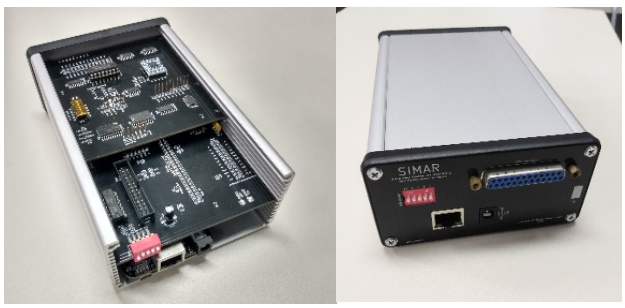


Figure 2: SIMAR base board and interface stack prototypes assembled, together, minimal unit installed in Sirius cabinets.

SPI and I²C Extender Modules

Another auxiliary device for SIMAR is a SPI extension, which aims to split and enable communication with up to six other devices. Device selection is performed by decoding four bits, which address the chip-select signal to the correspondent device. These bits get into the SPI expander as a configuration word.

Similarly, an I²C extender has been designed.

DATA ACQUISITION AND MONITORING

Acquiring Sensor Data

In order to increase system performance and allow it to work with multiple sensors, all coding is in C/C++ and Python modules may be created from these libraries.

With the base board only, SIMAR communicates through up to 4 different I²C channels, making it possible to monitor multiple cabinets with a single unit, greatly reducing costs. Switching is done through memory mapped input/output, diminishing switching times and increasing maximum polling frequencies. This strategy is also applied to other interfaces, such as SPI and digital inputs/outputs.

Currently, SIMAR supports the BMx280 family of sensors, automatically detecting and naming up to 8 different sensors (2 sensors per channel). Communication speed and sampling rates are limited by I²C channel switching delay and minimum reading/standby times for the sensors, which result in a maximum sample rate of approximately 800 Hz per sensor when all channels are active. If only one sensor per channel is active, it is possible to reach a theoretical limit of 1200 Hz for sample rates, given that only basic system and SIMAR processes are running on the main core.

For other sensors and auxiliary boards that require a faster communication solution, the BeagleBone's PRUmay also be used as some of its pins are mapped on SIMAR bus, allowing higher polling rates and predictability [3]. For example, it could count multi-purpose pulses, up to 10 MHZ, in 4 channels, while simultaneously performing other functions in the main core. Functions could be communicating with 8 different sensors, an external ADC, processing remote commands. One of the expected featuresconcerning PRUs and SIMAR is to measure duty cycles of a digital waveform in the order of tens of MHz.

Even though a containerized Redis database [4] is running in Controls cluster, latest values for any variable are stored on a local database running on each BeagleBone. Data transfer is fast, reliable and dumps to disk are easily configurable. Output signals are recovered from this database after either a cold or warm system boot. Yet, if needed, a copy of all logged sensor data is also written to any USB storage device connected to SIMAR automatically.

Deduction Algorithms for Door Status

Through the BMx280's family high accuracy and sensitivity to pressure changes (with its IIR filter disabled), it is possible to eliminate the need for a door status sensor such as a switch. By monitoring pressure moving averages, sustained and significant (> 0.3 hPa) pressure changes are counted as door status changes.

Thanks to the use of a Redis database, power outages and unplanned shutdowns do not affect the pressure moving average comparisons. When any of these situations happen, the moving average is automatically adjusted based on external pressure changes which accompany the expected pressure changes inside the cabinet since the last time it was connected. The system has shown itself to be resilient against false positives.

Wireless Connection

A very special SIMAR use-case, under software development and integration tests, is using a wireless connec-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV027

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

tion to share data values between sensors and a remote Redis database. In this case, SIMAR connects to the cluster and retrieves data from the containerized database, so that static BeagleBone IPs are not mandatory to access acquired data.

It also allows the possibility to have portable SIMAR units, which are powered and start data acquisition on startup and data streaming automatically once a compatible wireless network is available.

EPICS INTEGRATION

Sirius' Control System is EPICS based, which leads to the need to translate SIMAR Redis variables into EPICS Process Variables (PVs). The EPICS IOC for this application runs atop a custom auto-failover Asyn driver [5], developed in collaboration with Dirk Zimoch, from Paul Scherrer Institute (PSI). With this driver, it automatically switches to alternative servers in case of any disconnections, which is especially useful when coupled with high availability solutions, as a cluster or systems with hardware redundancy.

The IOCs are containerized and run on the Controls System cluster, querying Redis databases for data updates, utilizing a custom auto generated Redis IOC [6]. This approach reduces overheads resulting from hosting an additional communication channel for each SIMAR unit, since Redis is already present on the default Sirius BeagleBone image.

Performance-wise, the IOC solutions applied to SIMAR make for high-frequency polling that accompanies the performance of the main program running on all SIMAR units, reducing data losses. Utilizing Redis also allows holding the data for as long as possible while the IOC catches up, in case that ever occurs. The conceptual design of the software stack is shown in figure 3.

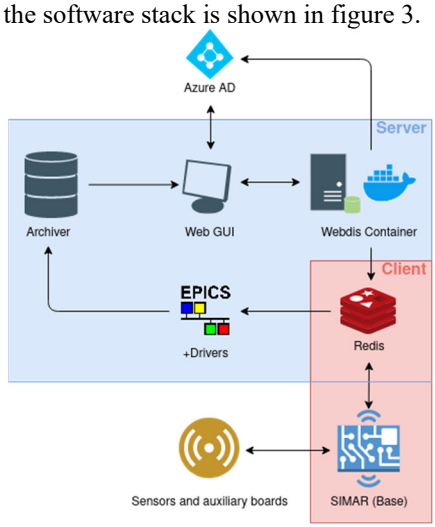


Figure 3: SIMAR's software stack.

USER INTERFACE

The user interface for SIMAR (figure 4) is web-based and responsive, allowing access from mobile devices and desktops alike. Based on the Vue framework, it features real-time data updates, graphical configuration menus, granular permissions, and safe authentication, based on Microsoft Azure's technology [7], which is the default authentication for CNPEM collaborators.

Fetching information is also trivial, since the designed GUI makes use of "Epics2Web" [8] API, utilizing web sockets to communicate in an efficient and quick manner with the EPICS server. In addition, the GUI supplies dynamic links to the EPICS Web Archiver Viewer, allowing users to quickly view historic data for each device.

Communication also occurs through a custom version of Webdis [9], an HTTP interface for Redis database. Added functionalities regarding default applications are server-side logging and authentication for additional security. In order to increase performance, Redis' script cache functionality is used, bundling together batch requests.

Authentication was added to restrict interaction with outlet actuation and custom alert limit functionalities. No user credentials are stored, and all communication is compliant with OAuth2and HTTPS standards, with only a token being received by the Webdis server, which is used to confirm the user's identity.

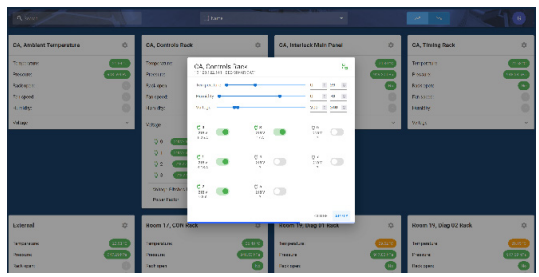


Figure 4: SIMAR's web-based graphical user interface.

In addition, in-house tools are available which allow users to be notified whenever a limit is exceeded, transmitting notifications through email and Telegram instant messages, for example.

INITIAL INSTALLATION

Currently, SIMAR prototypes have been applied to monitor some critical environments, such as 12 default cabinets from various subsystems in a high-density area subjected to higher ambient temperatures. A base board acquires data from sensors located inside each cabinet, positioned on the most critical region.

Figure 5: Sensor unit placed inside a cabinet.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV027

A wireless unit, initially connected to the corporate network due to connectivity availability, has been in operation for some months in order to acquire environmental data (ambient pressure, humidity, and temperature) outside the building for metrology refinements.

Dedicated cabinets (timing system, controls servers and interlock system) have recently started being monitored by SIMAR, acting as a replacement for a system that required physically dismounting the sensor to retrieve data and recharge batteries, figure 5 demonstrates a sensor installation in a cabinet. The graphs shown in figure 6, figure 7 and figure 8, exemplify how the environment data is being presented to the user.

Figure 6: Temperature measurement of special cabinets and environment temperature at Connectivity Area.

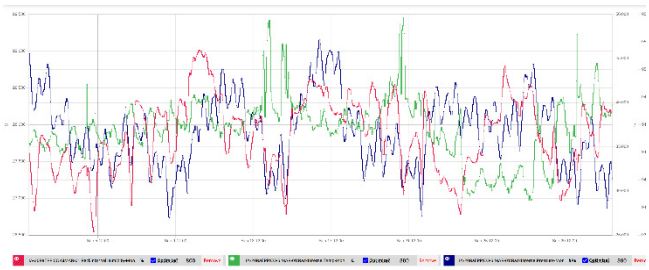


Figure 7: Temperature, pressure and humidity measurement for a pulsed-magnet cabinet.

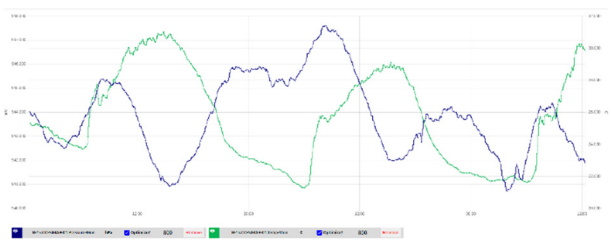


Figure 8: External (out of building) measurements, helping with subsystem stabilities.

The data provided by SIMAR allows new or existing HVAC installations to be tailored to the needs of each individual room, subsystem, or cabinet. They also help with narrowing down possible issues with airflow and humidity-sensitive equipment, as well as making it easier for operators to monitor engineering cabinets health and status. It is also important for detecting unusual temperature variations (both internal/external cabinets and outside the building), thanks to its high precision, data acquisition rate, ease of installation and high mobility.

FUTURE APPLICATIONS AND IMPROVEMENTS

Many improvements are planned, and some are already in progress. The next application under development considers power outlets switching and monitoring (voltage, current, frequency and power factor), which will greatly increase system added value and expand actuating possibilities.

Other features are also planned, such as sound signature analysis in order to predict and prevent failures more reliably, local alarms and support for more sensors. Powering through PoE would reduce the number of cables anda dedicated IoT wireless network would eliminate communication weak points.

CONCLUSION

As part of Sirius' engineering team effort to continuously improve and reach new standards of reliability, efficiency, and stability, SIMAR has demonstrated that it is quite essential for on analyzing environmental and operational data and act over it whenever it is necessary.

Such data can be used to predict failures, schedule preventive maintenance, and help members to planout upgrades in advance, while remote actuation provides a channel of interaction for otherwise inaccessible devices and systems. If environment data is acquired and all the necessary interfaces to actuate over it are available, analyzing that data along with specific subsystem variables might lead to a better environment that increases overall stability and reliability.

Although SIMAR is still in its commissioning phase, it is currently expected to be installed in all Sirius Engineering cabinets, along with other locations and systems as requested, since it is highly versatile, easy to set up and a low-cost solution compared to commercial ones.

These developments are part of the many steps to guarantee that Sirius' commissioning and operation are efficient, stable, and reliable, achieving the expected research output.

REFERENCES

[1] J. Franco *et al.*, "Software and Hardware Design for Controls Infrastructure at Sirius Light Source", in Proc. 17th Int. Conf. on Acc. And Large Exp. Physics Control Systems (ICALEPCS2019), New York, USA, Oct. 2019, pp. 263-267. doi.org/10.18429/JACOW-ICALEPCS2019-MOPHA031

[2] P. Nallin, J. Franco, R. Ito, A. Rodrigues, "Exploring Embedded Systems' Dedicated Cores for Real-Time Applications", in Proc. 17th Int. Conf. on Acc. And Large Exp. Physics Control Systems (ICALEPCS2019), New York, USA, Oct. 2019, pp. 1036-1040. doi.org/10.18429/JACOW-ICALEPCS2019-WEMPR003

[3] M. Watkins and C. Betancourt. Ensuring real-time predictability: Leveraging TI's Sitara Processors Programmable Real-Time Unit. TI Whitepaper, 2014.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV027

[4] S. Chen, X. Tang, H. Wang, H. Zhao, and M. Guo, "Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis," 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 1660-1667, doi.org/10.1109/TrustCom.2016.0255

[5] Asyn Failover, available in https://github.com/cnpem-iot/asyn-failover

[6] Redis IOC, available in https://github.com/cnpem-iot/redis-ioc

[7] V. Bertocci. "Modern authentication with Azure Active Directory for web applications", 1st ed., USA, 2016, Microsoft Press.

[8] R. Slominski, T. Larrieu. "Web Extensible Display Manager", in Proc. 17th Int. Conf. on Acc. And Large Exp. Physics Control Systems (ICALEPCS2017), Barcelona, Spain, Oct. 2017, pp. 852-856. doi.org/10.18429/JACOW-ICALEPCS2017-TUPHA181

[9] A Redis HTTP interface with JSON output, available in-https://github.com/nicolasff/webdis

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV028

CompactRIO CUSTOM MODULE DESIGN FOR THE BEAMLINE'S CONTROL SYSTEM AT SIRIUS

L. S. Perissinotto†, F. H. Cardoso, M. M. Donatti
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

The CompactRIO (cRIO) platform is the standard hardware choice for data acquisition, controls and synchronization tasks at Sirius beamlines. The cRIO controllers are equipped with a processor running a Real-Time Linux and an embedded FPGA, that could be programmed using Labview. The platform supports industrial I/O modules for a wide variety of signals, sensors, and interfaces. Even with many available commercial modules, complex synchrotron radiation experiments demand customized signal acquisition hardware to achieve proper measurements and control systems integration. This work aims to describe hardware and software aspects of the first custom 8-channel differential digital I/O module (compatible with RS485/RS422) developed for the Sirius beamlines. The module is compliant with cRIO specifications and can perform differential communication with a maximum 20 MHz update rate. The features, architecture and its benchmark tests will be presented. This project is part of an effort to expand the use of the cRIO platform in scientific experiments at Sirius and brings the opportunity to increase the expertise to develop custom hardware solutions to cover future applications.

INTRODUCTION

In a typical synchrotron beamline, a wide variety of electronic devices are employed for controlling and monitoring complex scientific experiments. In a general aspect, these equipment acts to control the beamline components such as shutters, mirrors, monochromators, diagnostic elements, motors, vacuum pumps, etc. These devices have different types of communication interfaces or electrical I/Os in different voltage levels or standards that must be proper integrated to the EPICS control system [1]. The CompactRIO (cRIO) [2] platform is the standard hardware controller used for data acquisition, control and synchronization tasks. The platform supports industrial I/O modules for manipulating a wide variety of signals and interfaces and despite the broad portfolio of commercially available I/O modules, custom hardware designs are often needed. To supply the demand for differential signaling communication, the first in-house C-series module was designed and comprises a 20 MHz max. update rate digital I/O compatible with RS485/RS422 [3] standard.

The hardware and software development were guided by the National Instruments cRIO-9951 [4] development kit, which provides guidelines for design implementation. The developed module will be used for long distance differential communication, triggering and synchronization. The knowledge applied in this development will be useful to develop custom hardware for future applications.

† lucas.perissinotto@lnls.br

HARDWARE

The module has 8 differential channels with configurable direction, compatible with RS485/RS422, and can achieve up to 20 MHz update rate. An external power supply may be required in some conditions, which depends on the channels' directions and data rate. Differential signals and external power supply are available to the user in the front panel through a 37-pin DSUB connector and it has two led indicators that shows the internal powering status and external powering needs. The module is connected to CompactRIO using a high density 15-pin D-SUB connector on the back of the board.

To safeguard the controller from ESD damage, insulation topology [5] was adopted using digital isolators between all internal and external signals to cRIO bus. The circuit is physically separated into the non-insulated side, where compactRIO is attached, and the insulated side where the application signals are available.

The module circuit is divided into three main blocks: power supply, transceivers, and control logic. The power supply block is responsible for insulating, protecting, and managing all power supply sources, internal and external. The transceivers block converts the internal bus single-ended signals into insulated external differential signals, or vice versa, depending on the configured direction. The control logic block identifies compactRIO control signals and implements all internal mode's support logic. The blocks organization in the PCB can be seen in Fig. 1.

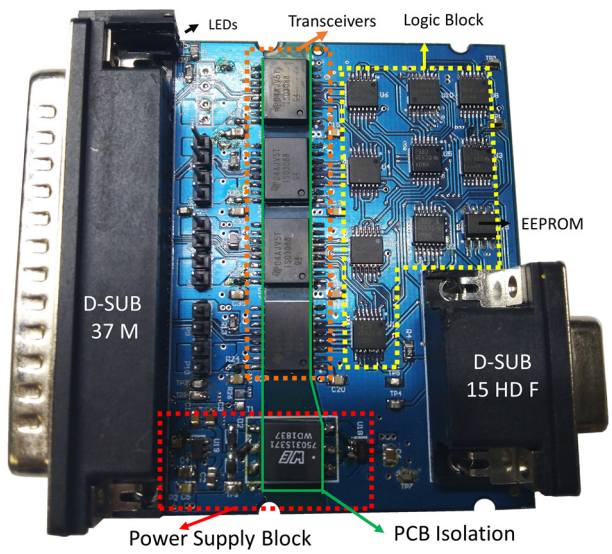


Figure 1: Device blocks on PCB.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV028

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

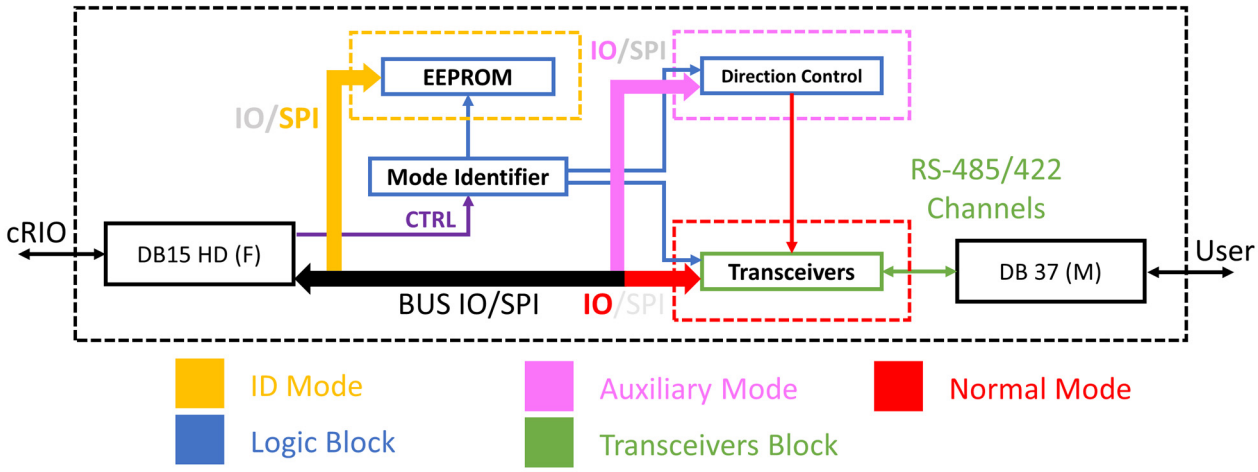


Figure 2: Device block diagram.

The module supports the four internal compactRIO operating modes [6]: Idle, Id, Auxiliary and Normal. In each mode, the bus is used to control a different interface except for the Idle mode, where all module interfaces are disabled. In Id mode, the bus is used to access the embedded EEPROM, allowing the module's identification procedure. The Auxiliary mode allow the user to configure differential channel direction. In Normal mode, the bus is used to access transceivers input or output data channels, depending on the direction set in Auxiliary mode. The operation modes diagram is shown in Fig. 2.

Transceivers

To obtain the required performance and protection, eight insulated half-duplex transceivers IC [7] is used. The transceiver converts single-ended (TTL) signals to differential (RS485) standard with a maximum 20 MHz update rate. Each transceiver's channel is connected directly to the compactRIO bus.

The transceivers have two pins for data signals, one for input (R) and another for output (D). Since the transceivers are half-duplex, both data signals tied together to reduce the number of demanded digital I/O from the controller. This arrangement can be seen in Fig. 6.

In Normal mode, the transceivers have their inputs and outputs set as pre-defined in Auxiliary mode. When other modes are selected, all data pins from the transceivers switch to high impedance state, granting the bus the ability to be used in module identification procedure or channels direction configuration. The differential side are also switched to high impedance to prevent any damage during module initialization.

Regarding the physical aspects of the transceivers block, the differential signals were routed as a pair and designed to have 120 ohms characteristic impedance. The differential traces have termination resistors, that can be enabled by manual jumpers, to provide matched load for tests and specific applications.

Control Logic

The logic block is responsible for identifying each compactRIO internal operation mode and controlling all interfaces that access the bus. The interfaces include the EEPROM, direction control and transceivers. All logic was implemented using logic gates from the 74 IC family. The logic implementation for internal modes identification is shown in Fig. 3.

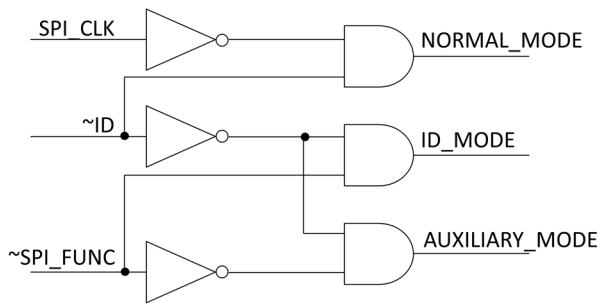


Figure 3: Internal operation mode logic identifier circuit.

The signals ~ID, SPI_FUNC and SPI_CLK comes directly from compactRIO bus in which the first two are intrinsically planned for this purpose. There is no signal intended to determine Normal mode, thus SPI_CLK was selected to be used as a digital I/O since SPI isn't used in this mode.

The EEPROM logic was implemented using the Id mode signal and the chip select SPI signal, as can be seen in Fig. 4. EEPROM SPI signals share the bus with I/O's used in Normal and Auxiliary mode. For this reason, the EEPROM SPI interface is activated only by control signal ~CS during Id mode.

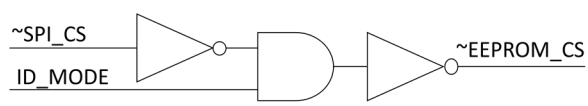


Figure 4: EEPROM ~CS logic control.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV028

An 8-bit shift register [8] with output implemented by flip-flops is used to set and store the data channels logic state directions in Auxiliary mode. Auxiliary mode signal provides access to the shift register for three digital I/O's from the bus. Figure 5 shows the logical scheme implemented to store channels direction that are used in Normal mode.

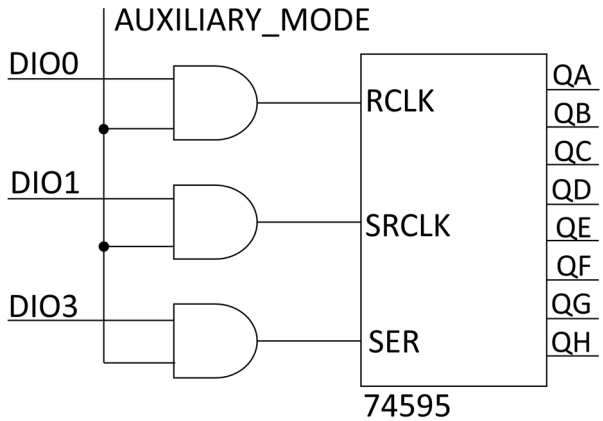


Figure 5: Shift Register and control logic.

The transceiver, on the non-insulated side, has different input and output data pins. Each data pin has its configuration pin, which configures channels direction and high impedance state. Both data pins have been tied together to reduce the need of extra digital I/Os from the bus. The implemented control logic for configuration pins allows it to be used in both directions during normal mode and keeps the outputs at high impedance in other modes. The logic implemented to control each transceiver can be seen in Fig. 6.

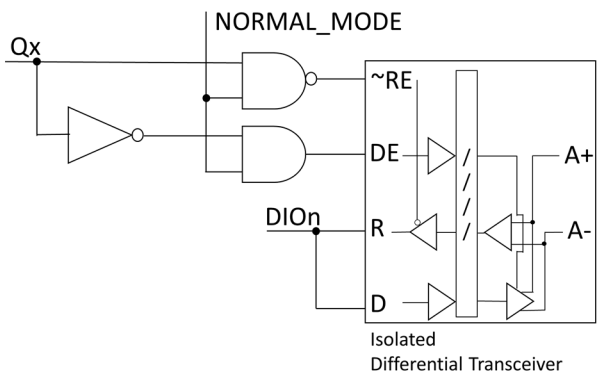


Figure 6: Transceiver control logic.

Power Supply

The power supply block was carefully designed to attend the voltage and current specification and provide protection for the external and internal circuitry. When internal power is used (from cRIO controller), isolators and current limiters are used to guarantee that compactRIO is protected from damaged caused by short-circuit, overcurrent or electrostatic discharge.

The current limiter was placed on the isolated side, thus an overcurrent or short-circuit would not disable the logic block on the non-insulated side. National Instruments documentation restricts the bus current draw to 200 mA.

The current consumption of the non-insulated side was estimated at 45 mA, consequently, the non-insulated side could draw a limit of 155 mA. Considering the transformer conversion factor (1:1.1) and the conversion efficiency (90%), the limiter was configured to limit current draw to 120 mA, in order to limit the input module current in 190 mA. The power supply block diagram can be seen in Fig. 7.

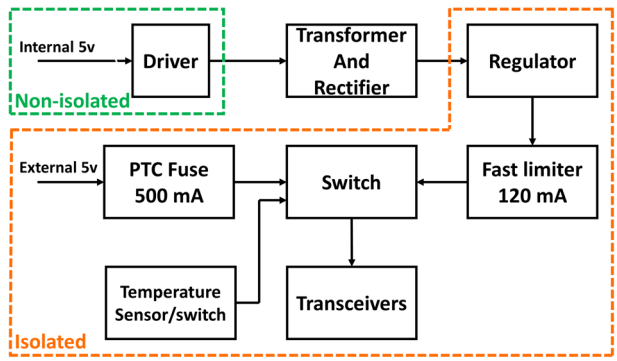


Figure 7: Module's Power Supply.

The internal power supply is adequate for some use conditions, but often an external power supply will be necessary. An electronic power switch is used to manage the use of external/internal power supply. A 500 mA PTC fuse protects the input from short circuits. Two led indicators (LED's) are present in the front panel: a red light indicates the limiter is activated, so external power supply is needed; a green light indicates the module is internally powered.

The external power supply is needed when current consumption exceeds the 200mA bus limit, but the internal module total power dissipation could be monitored to avoid temperature rising. The design rules limit the internal power dissipation in 1.5 W and for all transceivers used at same time (at the highest update rate) the board dissipates nominally 2.4 W. A temperature sensor and switch were used to monitoring the internal temperature in the maximum dissipation power condition and turn off the external power supply in case temperature rise above 60 ºC. The temperature monitor is configurable and can be used in any desired protection logic defined by the user.

SOFTWARE

The module software supports comprises the development of configuration files (XML) and Labview codes (VI). The elaboration of such items involves a series of steps and development modes, and all must be done using Labview environment and is detailed described in the development kit software documentation.

CompactRIO users interact with modules in FPGA environment [9] through Nodes (specific labview block) [10], which types can be I/O's, Methods or Properties. Each module Node usually has codes scripted behind them or can be mapped directly to digital I/O available in compactRIO (D-Sub-15-HD).

The strategy adopted to achieve an update rate close to FPGA clock speed was to map the digital I/O's behind I/O Node. As described in the transceiver's subsection, the controller digital I/O's are connected to the module's

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV028

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

internal bus just when the module is in Normal Mode. The differential signal works as an extended digital I/O's mapped in the I/O Node. This arrangement can be seen in Fig. 8.

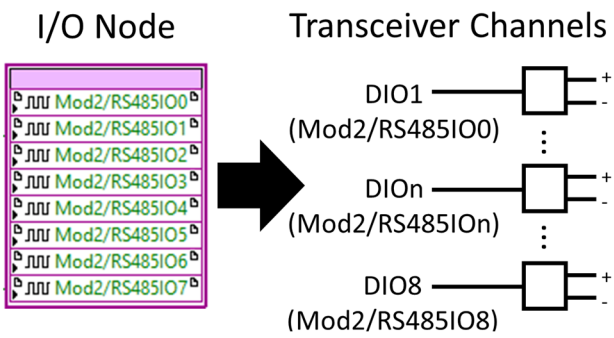


Figure 8: I/O's Node mapped direct in module bus.

In module support block, there is the module core code that works as a "main" function in a textual programming language. The core code executed in this module is responsible for module initialization and channel direction change procedure. This code runs in a loop, waiting for a direction change to be indicated by a Method Node. When a Method Node is used, a byte is passed as an argument, and each bit has, respectively, a differential channel associated, as shown in Figure 9.

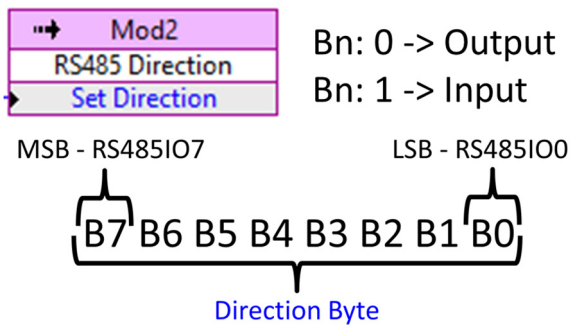


Figure 9: Change direction Method Node.

The core code switches the module to Auxiliary mode at the beginning of each channel direction change procedure. It transfers a byte (bit-by-bit) the to the shift register over DIO3 during the rising edge of DIO1. In the end, the shift register output is updated by a rising edge on DIO0, and the module returns to Normal Mode. The block diagram of direction change is shown at Fig. 10.

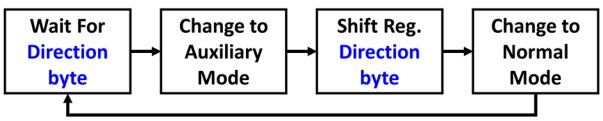


Figure 10: Change direction core code diagram.

TESTS

The module was initially tested using an in-house developed test jig based on LPC1768 [11] microcontroller. All compactRIO modes, functions and I/O's were emulated and tested before board connection at cRIO data bus. With a mature software version ready, performance, consumption and temperature tests were performed.

Performance tests were conducted and the module was able to reach the maximum specified 20 MHz output data rate, as shown in Fig. 11. Communication tests were performed using three strategies: a loopback connection, a commercial NI-9753 [12] module and another developed prototype. In these communication tests, long cables (20 meters) and termination resistors (120 ohms) were used with all simultaneous channels transmitting or receiving data in both directions.

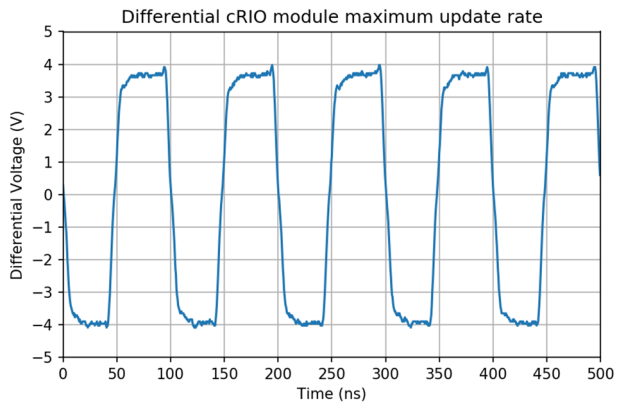


Figure 11: Module maximum update rate output waveform.

The current consumption was measured with the channels configured as input and output. In the input direction test, the module was supplied through the compactRIO connector. Differential signals were applied in the external connector by another module, while the current and limiter indication were verified for different update rates. The results are shown in Table 1.

Table 1: Module Internal Current Consumption

Update Rate Input (MHz)	Internal Current (mA)	Limiter indicator
20	215	1
10	190	0
5	179	0
2.5	173	0
1.25	170	0
0.1	167	0

For the transmitting data (output direction), the module was powered by both sides (internal and external power supply), and the current consumption was measured in the external connector for different data rates. The results are summarized in Table 2.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV028

Table 2: Module External Current Consumption

Update Rate Input (MHz)	External Current (mA)
20	403
10	372
5	342
2.5	311
1.25	280
0.1	169

The maximum measured consumed power, in the transmission test (output direction) was 2W. During this test, the internal current was not monitored but is estimated to be less than half of the values presented in Table 1. In this case, the total dissipated power reaches approximately 2.5W, above the stipulated limit at design rules. The effects of excess power dissipations were verified by keeping three adjacent modules under such conditions, while the temperature was being monitored until stabilization. The observed module temperature was 47.5ºC, working at the room temperature about 23±1ºC and stabilization took two hours.

CONCLUSIONS

The first custom 8-channel differential digital I/O module (compatible with RS485/RS422) in a C-series form factor for the Sirius beamlines was designed. Using high-performance isolated half-duplex differential line transceivers, the module can perform differential communication with a maximum 20 MHz data rate. The module is pin-compatible with the commercial NI-9753 module [12], and it has eight channels independently configurable as input or output. A selectable 120 Ohms resistor is placed to guarantee proper termination in receiver mode.

The designed module is compliant with NI cRIO specification in certain conditions and it has some operational limitations related to the current draw, which can be overcome by the use of external power supplies.

In terms of power dissipation, the module could exceed the defined 1.5 Watts limit in some situations. When the cRIO crate is fully populated, the module temperature must be monitored, and the use of temperature interlock must be evaluated.

Due to the use of low-cost transceivers, the module was able to be manufactured for less than one-tenth the price of the commercial benchmark, which makes it a cost-effective alternative for a wide variety of applications, in special for long-distance digital signal transmission.

FUTURE PLANS

This project is part of an extensive effort to expand the use of the cRIO platform at Sirius beamlines. The developed prototype fulfilled its objective, allowing the developers to obtain the necessary knowledge in hardware and software implementation. The experience acquired by the development team will contribute for the design of more complex modules. Future developments aim to provide solutions for high-resolution digitizers, low latency/low jitter triggering modules, and other scientific experiment-oriented demand.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the Brazilian Ministry of Science and Technology as funding agency and all the engineering and science teams members working for the Sirius project.

REFERENCES

[1] EPICS, https://epics-controls.org/

[2] National Instruments, CompactRIO Platform Overview, https://www.ni.com/pt-br/shop/compactrio.html

[3] M. Soltero *et al.*, RS-422 and RS-485 Standards Overview and System Configurations, SLLA070 Application Report, https://www.ti.com/lit/an/slla070d/slla070d.pdf

[4] National Instruments, cRIO-9951, https://www.ni.com/pt-br/support/model.crio-9951.html

[5] Texas Instruments, How to Isolate Signal and Power for an RS-485 System, Oct. 2018, https://www.ti.com/lit/an/slla416c/slla416c.pdf

[6] National Instruments, NI cRIO-9951, CompactRIO™ Module Development Kit User Manual, Software User Manual, Nov. 2017, https://www.ni.com/pdf/manuals/375951b.pdf

[7] Texas Instruments, ISO3088 Datasheet, https://www.ti.com/lit/ds/symlink/iso3088.pdf

[8] Texas Instruments, SN74HC595 Datasheet, https://www.ti.com/lit/ds/symlink/sn74hc595.pdf

[9] National Instruments, FPGA Module, https://zone.ni.com/reference/en-XX/help/371599P-01/lvfpgamain/lv_fpga_main_title/

[10] National Instruments, FPGA I/O Node, https://zone.ni.com/reference/en-XX/help/371599P-01/lvfpga/fpga_io_node/

[11] NXP, Arm LPC1768 Board, https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc1700-cortex-m3/arm-mbed-lpc1768-board:OM11043

[12] National Instruments, NI-9753 information page, https://www.ni.com/pt-br/support/model.ni-9753.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV031

STATUS OF THE uTCA DIGITAL LLRF DESIGN FOR SARAF PHASE II

J. S. Fernandez, P. Gil, G. Ramirez, Seven Solutions, Granada, Spain
G. Ferrand, F. Gohier, G. Desmarchelier, N. Pichoff, CEA, Saclay, France

Abstract

One of the crucial control systems of any particle accelerator is the Low-Level Radio Frequency (LLRF). The purpose of a LLRF is to control the amplitude and phase of the field inside the accelerating cavity. The LLRF is a subsystem of the CEA (Commissariat à l'Energie Atomique) control domain for the SARAF-LINAC (Soreq Applied Research Accelerator Facility – Linear Accelerator) instrumentation and Seven Solutions has designed, developed, manufactured, and tested the system based on CEA technical specifications. The final version of this digital LLRF will be installed in the SARAF accelerator in Israel at the end of 2021. The architecture, design, and development as well as the performance of the LLRF system will be presented in this paper. The benefits of the proposed architecture and the first results will be shown.

INTRODUCTION

The SARAF-LINAC project intends to accelerate proton and deuton beam currents from 0.4mA to 5mA up to 40MeV for deutons and 35 MeV for protons. To achieve this, the field in the cavities needs to be controlled and regulated. The LLRF is the device in charge of maintaining the cavity gradient and phase stability in presence of beam. Seven Solutions has designed and implemented the LLRF system including all the hardware, gateware (FPGA code) and software needed to fulfil the specifications derived from [1]. The LLRF channels required for the SARAF-LINAC phase II [2] will drive both normal conducing cavities and superconducting cavities.

The factory acceptance tests (FAT) results will be detailed in this paper.

SPECIFICATIONS

The LLRF must be able to regulate the cavity field both in pulsed mode and in continuous wave (CW). From the simulations and studies performed in [1], the main LLRF system requirements were derived. They are listed in Table 1. The frequency of the full accelerator is 176MHz. This is the reference frequency for the LLRF system.

Table 1: LLRF Requirements

Requirement	Value
Operation frequency range	176MHz +/-100KHz
LLRF delay	< 1us
Input amplitude RMS error	< 0.03%
Input phase RMS error	< 0.03º
Output amplitude stability	< 5%
Output phase stability	< 5º

HARDWARE

The LLRF HW is based on the uTCA.4 technology. The system is composed by two boards (Fig. 1):

The LFE (LLRF Front End) is in charge of conditioning the RF inputs signals to interface them to the ADCs. In addition, the RF drive outputs are amplified and filtered in this board.

The AMC (Advanced Mezzanine Card) digitizer controller board includes all the components (ADCs, DACs, PLL, FPGA, DDR memory…) needed to perform the acquisition and generation of the RF signals as well as all the digital signals processing to implement the different algorithms and features included in a LLRF system. The design based on a FPGA provides the system with a great flexibility making it capable of being adapted to the requirements of different accelerator facilities and to be placed on a rack or in a single standalone mode.

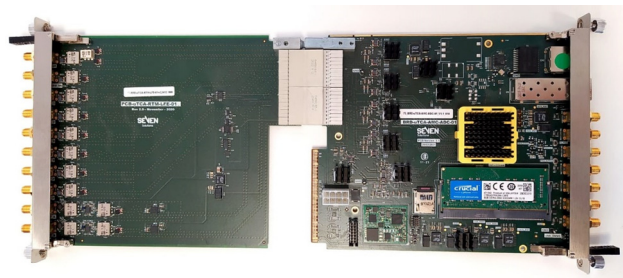


Figure 1: LLRF HW composed of LFE and AMC boards.

The HW is designed to have two LLRF channels making possible to control two cavities with one set of board. Each channel generates one RF drive signal (Uamp) and monitors three RF signals (Fig. 2):

- Ucav: Cavity voltage.
- Uci: Incident voltage.
- Ucr: Reflected voltage.

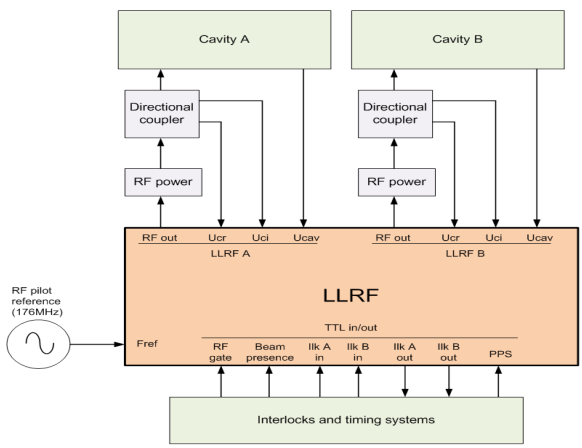


Figure 2: LLRF block diagram.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV031

GATEWARE

The LLRF system is based on SoC technology. A Zynq UltraScale FPGA from Xilinx has been used in the design. This family of FPGAs combines a great amount of programmable logic and two dual core ARM processors in the same chip, offering enough resources and flexibility to implement all the functionalities that characterize a LLRF system.

A block diagram with the main gateware elements is shown in Fig. 3.

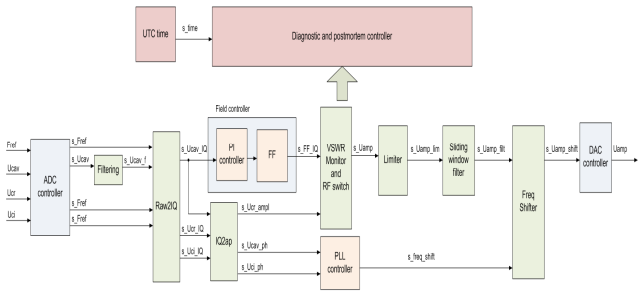


Figure 3: Gateware architecture block diagram.

The LLRF system can operate both in pulsed mode and in continuous wave. Moreover, it can operate in opened loop for conditioning and in closed loop for cavity regulation.

The main functionalities implemented in the FPGA are:

- ADC controller: In charge of configuring the ADC chips and acquiring the RF samples from them.
- Demodulation: To get the IQ component of the RF signal form the ADCs.
- Calibration: Used to compensate gain and phase offsets in the acquired RF signals.
- Field controller: Composed by a PI (Proportional-Integral) controller, to maintain stable the amplitude and phase of the cavity field, and a feedforward controller to compensate the beam loading effects.
- VSWR monitor: To monitor the reflected voltage and shut off the RF generation in case this voltage exceeds a configurable limit.
- Limiter: To prevent from generating undesired high RF output levels.
- Sliding window filter: To perform pulse shaping over the RF output pulses.
- Freq shifter: Used to cause a frequency displacement over the RF output signal.
- PLL controller: Allows to track the cavity resonant frequency automatically when the LLRF is operating in opened loop.
- DAC controller: Responsible for configuring the DAC chips and sending the data to be converted to analog domain.
- Diagnostic and post-mortem: The systems store the value of the RF signal as well as other internal signals of interest and the status of the system in the DDR memory. These data are used for displaying in real time some information. In addition, postmortem files are generated, in case of alarm, which allow the operator to investigate the causes of the alarm and to know the status of the LLRF at that time.

EMBEDDED SOFTWARE

Figure 4 illustrates the main architecture developed on the Zynq UltraScale to provide the functionality and communications required for the control system installed on the LLRF system.

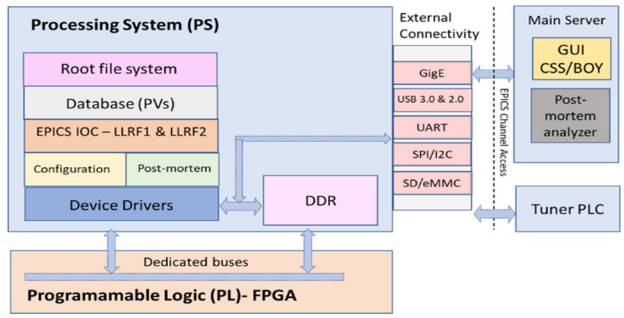


Figure 4: Software architecture.

The presented architecture consists of:

- Operating system: generated using the open-source tool Buildroot. Generates a single image to be loaded on the board contain the Linux kernel image, root file system, BOOT.bin (FPGA image), device tree and uboot.
- Low level drivers: allows the communication between the control system layer and low level devices. Communication between FPGA, DDR memory and I2C/SPI devices, such as EEPROM, are part of the drivers included in the embedded software.
- Control system layer: allows the communication between the low level drivers and the EPICS Channel Access. EPICS based is used in this development, and more specifically, AsynPortDriver is used as interface to the EPICS Channel Access (CA). Records and PVs are defined to access to the different elements defined to monitor and control the LLRF.

OPERATOR INTERFACE

The operator interface is implemented in CSS (Control System Studio) and Python. It enables the user to easily control the system making used to the interface provided by EPICS with the process variables (PV). Figure 5 is an example of how to control tasks such as monitoring, settings and display of PVs on a very easy way.

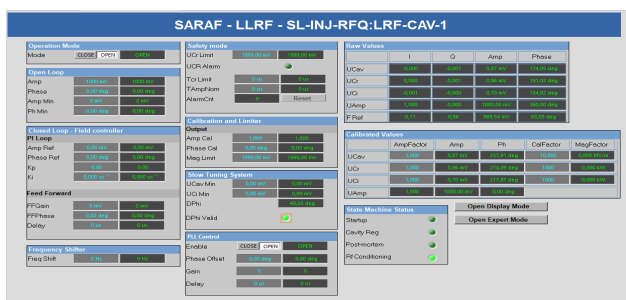


Figure 5: Graphical user interface.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

FAT TESTS RESULTS

In this section, some of the main tests performed in Seven Solutions laboratories are described.

Output Jitter

The jitter is directly related to the system stability. During the FAT, the jitter of the output signal Uamp has been measured. Figure 6 shows that the LLRF system has a very low additive jitter. The output jitter is 182fs when operating with a frequency reference source with 112fs of rms jitter.

Figure 6: LLRF output jitter.

Stability in Opened Loop

The RF output stability has been measured for amplitude and phase for 15 minutes. The measures have been taken when the LLRF operates in opened loop. The results are shown in Table 2.

Table 2: Output Stability

Magnitude	Value
Amplitude stability	0.014%
Phase stability	0.315º

Stability in Closed Loop

The stability in closed loop has been measured keeping the LLRF operating in closed loop during 48h. In Fig. 7 it can be seen the amplitude and phase of Ucav and the temperature fluctuation during the test. The results can be found in Table 3.

Table 3: Stability in Closed Loop

Magnitude	Value
Amplitude stability	0.066%
Phase stability	0.034º

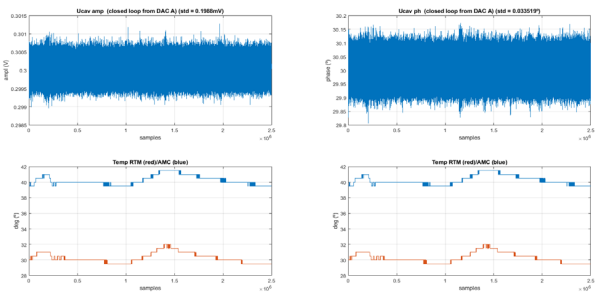


Figure 7: Long-term stability in closed loop.

Precision in Opened Loop

The precision in the acquisition of the input signal has been measured. For an input signal of 0dBm, the data acquired for the three inputs (Ucav, Uci and Ucr) have been stored during 300ms. The precision in amplitude and phase is shown in Table 4.

Table 4: Precision in Openend Loop

Magnitude	Value
Amplitude precision	0.03%
Phase precision	0.022º

Precision in Closed Loop

To measure the precision in closed loop the LLRF systems is connected in loop, that is, the output Uamp is connected to the input Ucav. With the system operating in closed loop (keeping the Ucav voltage in amplitude and phase to the values configured from the interface as setpoint), the output Uamp is stored and processed to get its stability.

As it is a precision measure, the capture is performed over a short period of time, in this case 1.2ms. The results are shown in Table 5.

Table 5: Precision in Closed Loop

Magnitude	Value
Amplitude precision	0.04%
Phase precision	0.018º

Loop Delay

One of the requirements was that the total loop delay of the LLRF system must be less than 1us. This delay is a measure of the response time of the PI controller when a disturbance occurs in the cavity field. To measure this, the LLRF is connected in loop and operating in closed loop. During the test, the proportional constant Kp of the PI controller is set to a value that causes the loop to diverge. In Fig. 8 can be seen the steps caused by the action of the PI controller. The length of these steps is the time the LLRF takes to perform an action when the measured signal (Ucav) changes. The measured response time is 912ns.

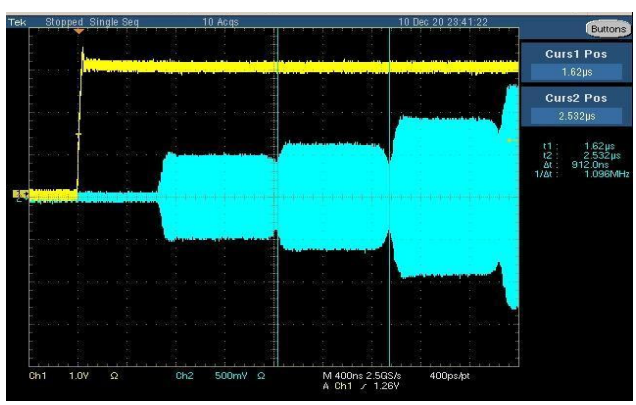


Figure 8: Loop delay test.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV031

Functional Tests

Several functionality tests have been carried out to check the different features that the LLRF systems implements.

A 176MHz bandpass filter with high Q factor (13000) has been used to test some features in conditions like those when the LLRF is connected to a cavity. In Fig. 9 it can be seen the LLRF regulating the voltage when the RF output (Uamp) is connected to the bandpass filter and the filter output is connected to Ucav. The LLRF is operating in closed loop and pulsed mode (RF gate in pink in the figure). In yellow we can see Uamp and in blue Ucav (the output of the bandpass filter).

The feedforward capability has been also tested during the FAT. In Fig. 10 it is shown RF gate signal in blue and the beam presence gate signal in yellow. As the LLRF is no connected to a cavity and there is no beam, it is observed how the Ucav signal increase its level, with the rising edge of the beam presence gate signal, but immediately the PI loop compensate this increase. In the same way, when the feedforward stops having effect, Ucav suffers a level drop that is quickly compensated by the PI controller.

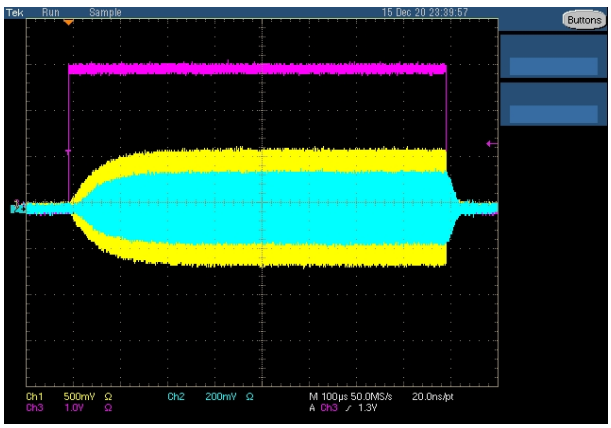


Figure 9: LLRF Ucav regulation test.

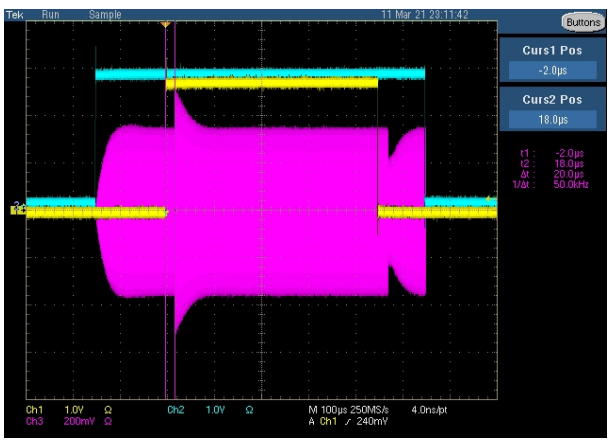


Figure 10: Feedforward capability test.

One of the most interesting functionalities of the LLRF system is its PLL functionality. Thanks to this functionality, the LLRF can track, when operating in opened loop, the resonant frequency of the cavity.

Two measurements of the filter phase characteristic were done separately. The first test, blue curve in Fig. 11, is the following: the incident voltage (Uci) and the output frequency of the LLRF are fixed. The transmitted voltage (Ucav) is measured (in mV), as well as the phase shift between Uci and Ucav. It gives a first graph of the phase as a function of the transmitted voltage for different frequencies. Then, instead of fixing the frequency, we did a second test by fixing the phase between Uci and Ucav. The PLL should lock to the frequency that provides the phase. This is the red curve in Fig. 11. In principle, if the PLL locks to the phase as expected, red and blue curves should be identical. This is precisely what was measured, validating the PLL function.

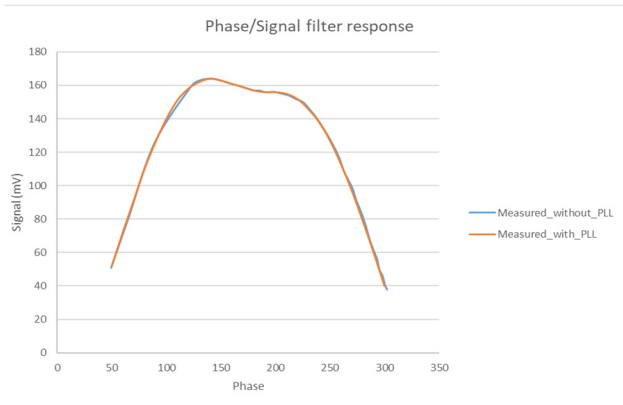


Figure 11: PLL functionality test.

CONCLUSIONS

This paper presents the LLRF system for SARAF project phase II designing and manufactured by Seven Solutions according to the specifications given by CEA. The FAT tests have ben satisfactory, and the next step will be the integration of the LLRF systems (18 resulting in a total of 36 channels) in the SARAF facilities in Israel.

REFERENCES

[1] L. Zhao, G. Ferrand, F. Gougnaud, R. Duperrier, N. Pichoff and C. Marchand, "Status of the LLRF system for SARAF project Phase II", arXiv:1909.12562 [physics.acc-ph] (2019).

[2] N. Pichoff, D. Berkovits, R. Duperrier, G. Ferrand, B. Gastineau, F. Gougnaud, M. Jacquemet, J. Luner, C. Madec, A. Perry *et al.*, "The SARAF-LINAC Project 2019 Status", in *Proc. IPAC'19,* Melbourne, Australia, May 2019, pp. 4352-4355, doi:10.18429/JACoW-IPAC2019-THPTS116

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV033

ARCHITECTURE OF A MULTI-CHANNEL DATA STREAMING DEVICE WITH AN FPGA AS A COPROCESSOR*

J. M. Nogiec†, P. Thompson, FNAL, Batavia, IL 60510, USA

Abstract

The design of a data acquisition system often involves the integration of a Field Programmable Gate Array (FPGA) with analog front-end components to achieve precise timing and control. Reuse of these hardware systems can be difficult since they need to be tightly coupled to the communications interface and timing requirements of the specific ADC used. A hybrid design exploring the use of an FPGA as a coprocessor to a traditional CPU in a dataflow architecture is presented. Reduction in the volume of data and gradual transitioning of data processing away from a hard real-time environment are both discussed. Chief design concerns, including data throughput and precise synchronization with external stimuli, are addressed. The discussion is illustrated by the implementation of a multi-channel digital integrator, a device based entirely on commercial off-the-shelf (COTS) equipment.

INTRODUCTION

One of the typical dilemmas designers face when building a system is whether to buy or build its components. This also applies to test and measurement systems used in High Energy Physics. Readily available, so-called commercial off-the-shelf (COTS) components are ubiquitous and can reduce development cost and offer product support by manufacturers. Unfortunately, the offered instruments or modules may not be precisely what is needed. A solution can be to build an instrument in-house, but from available COTS subcomponents, as is the case for the digital integrator device discussed in this article.

Construction of a multi-channel streaming integrator from COTS subcomponents is presented in the context of using an FPGA as a coprocessor in order to provide predictable and guaranteed data processing performance. The discussed integrator is functionally extensible, thus it has been named the Extensible Digital Integrator (EDI).

Digital integrators have been very successful in testing accelerator magnets, and they are crucial parts of rotating coil systems and single stretched wire systems, the cornerstones of the measurement toolset used in this domain.

FPGA AS COPROCESSOR

A Field Programmable Gate Array (FPGA) allows engineers to design and program data acquisition and control hardware to cater to the specific needs of their applications. Typically, systems employ FPGAs as controllers of hardware and signal preprocessors or conditioners. In these architectures, the FPGA is positioned between the I/O and the higher layers of data processing or control (see Fig. 1a).

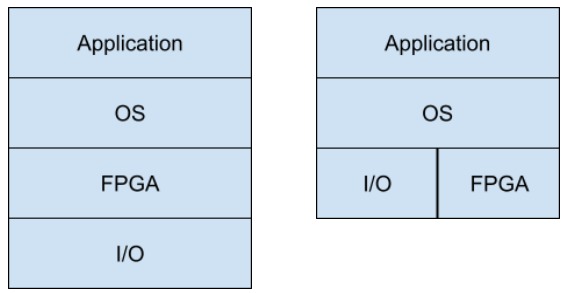


Figure 1: Layers in the system using an FPGA: a) for signal acquisition, b) as a coprocessor.

Another typical application of FPGAs is as a coprocessor offloading computationally intensive functions from a main processor. This solution offers deterministic performance in demanding data processing situations, especially in real-time applications (see Fig. 1b).

The EDI combines both of these hardware organizations into a single architecture. Here, the FPGA serves as both a coprocessor integrating signals and a real-time signal processor acquiring external triggering signals.

The performance of the analog-to-digital converters in dedicated PXI coprocessor modules is inadequate for our needs, and there is no access to the selected DSA ADC boards directly from the FPGA. These factors contributed to some design specifics of the EDI solution.

INTEGRATORS

Digital integrators have proven to be useful in building test systems to measure magnetic fields in accelerator magnets. In fact, they are often a crucial instrument in systems based on the rotating coil and single stretched wire techniques.

These instruments integrate input signals (voltages) over time intervals provided by internal or external triggers, such as a train of pulses or the output of an angular encoder. Historically, these systems evolved from voltage-to-frequency converters connected to up-down counters and trigger boxes [1, 2] to systems using analog-to-digital converters coupled with DSP coprocessors performing integration [3-8]. More recently, FPGAs have offered a very attractive alternative to other integrator hardware

* Work supported by the U.S. Department of Energy under contract no. DE-AC02-07CH11359
† nogiec@fnal.gov

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV033

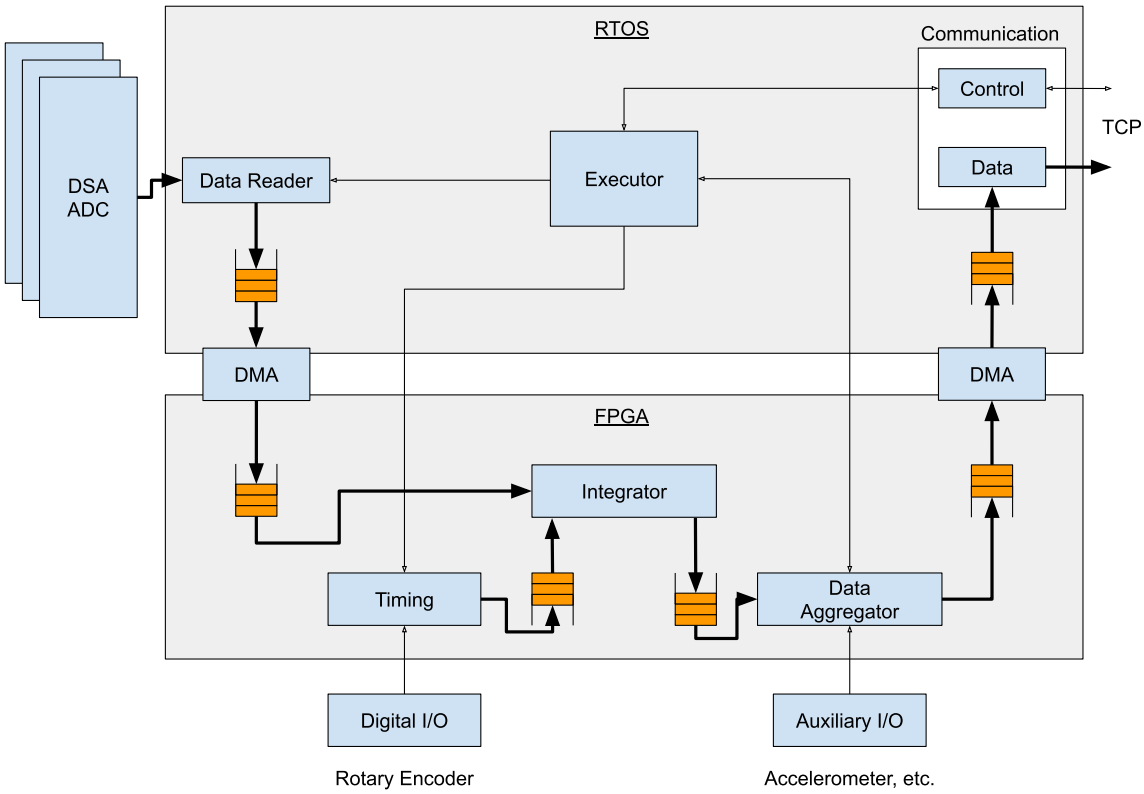


Figure 2: Organization of the EDI.

solutions, allowing for both acquisition of signals and signal processing [9, 10].

EDI HARDWARE

The EDI uses an FPGA for acquisition of external timing signals and a PXI bus for acquiring data from a set of analog-to-digital converters. Both the ADCs and the FPGA come as COTS PXIe modules produced by National Instruments Corporation.

FPGA

The PXIe-7856 reconfigurable I/O module featuring a Xilinx Kintex-7 FPGA provides ample logic area for creating integration and data processing algorithms. Memory space for queues is supplied in part by its 11,700 kbits of block RAM [11]. When used with a compatible PXIe chassis, these FPGA cards as well as the ADC cards will automatically synchronize with the 100 MHz chassis backplane clock with less than 250 ps of skew between modules [12]. Therefore, the ADC sampling clock and the 40 MHz FPGA clock used for keeping time between encoder triggers are derived from the same time base, which effectively eliminates clock drift between the modules.

Additionally, some ancillary functions (e.g., accelerometer readings for probe orientation monitoring) are provided by some built-in 16-bit successive approximation register (SAR) ADCs [13].

ADC

The PXIe-4464 ADC cards are members of the Dynamic Signal Acquisition (DSA) family of modules from National Instruments. With programmable gains and good frequency domain performance, these cards are well suited for harmonics measurement applications. The sampling rate is also adjustable, with a maximum of 204.8 kS/s. A CMRR of up to 105 dBc (depending on selected gain) provides adequate noise rejection on long probe cable runs. Samples are provided by a delta-sigma ADC with a resolution of 24 bits [14].

However, one potential drawback of this card is that it is not compatible with peer-to-peer streaming, necessitating the routing of sample data through the RT controller (adding some overhead) before it can be sent to the FPGA.

EDI ORGANIZATION

The programmatic implementation of the EDI includes software in the real-time operating system (RTOS) layer and firmware in the FPGA layer (see Fig. 2). Both the software and firmware are built as a set of collaborating modules communicating via queues.

Commands directing the operation of the EDI come via a TCP connection and are encoded by the control and communication module and are then sent to the executor module. The executor module keeps the state of the integrator and orchestrates the operation of other software and firmware modules.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The input data reader module starts data acquisition in the ADC channels and reads the incoming data, which subsequently are sent via a DMA channel to the FPGA where they are appended to the raw data queue.

At the same time, the timing module starts to read the triggering signals and for each trigger appends an element to the timing queue. The element contains the time elapsed from the previous trigger (integration time).

The integrator module reads an element from the head of the timing queue, calculates the number of ADC samples to integrate, performs integration, and appends the results of integration and integration time to the integrator output queue.

Depending on the configuration, a given number of elements in the integrator output queue are combined in the data aggregator module and sent via a DMA channel to the RTOS layer.

The integrated and aggregated data are received by the RTOS and are passed to the data communication module, which send them via a channel in the TCP-based connection to the consumer of the data.

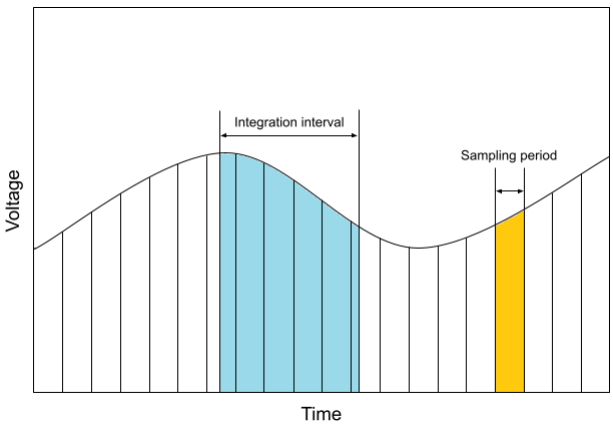


Figure 3: Integration interval and ADC sampling time.

INTEGRATION

The process of integrating signals over a given period of time is conducted in the FPGA. In this process, the samples from a given time interval are added up. The signals/pulses that trigger the starting and stopping of each integration interval come asynchronously to samples produced by the analog-to-digital converters. This necessitates the inclusion in the integration calculations the fractions of the first and last sample in the integration window (see Fig. 3). The beginning and end of each integration window is calculated using the sampling frequency and the integration time. The integrator module maintains a timeline for the sequence of samples that allows it to determine the number of samples (with the precision of a fraction of a sample) to be removed from the input queue and summed up.

The triggering information and the integration timeline are common to all data channels and the same operations are performed on each signal channel for each integration window. The original raw data stream is significantly reduced in size due to this integration process.

The implementation of integration has been conducted while paying close attention to the precision of calculations.

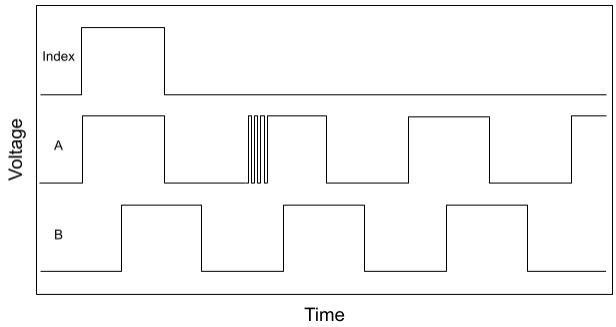


Figure 4: Sample angular encoder signals. Note noise example on A signal.

TRIGGERING

There are two main types of integrator triggering: internal and external.

Internal triggering relies on the FPGA clock to specify integration windows. The clock ticks are counted and after a required number is reached a new element is added to the timing queue.

External triggering is based on monitoring digital inputs for changes in their levels. A simple version will use a single digital input, providing a train of pulses controlling integration intervals. A more complex method, normally used in rotating coil measurement systems, is to trigger using angular encoder signals (see Fig. 4). Here, three signals are monitored to detect changes of their levels: index, A, and B. The A and B signals have their phases shifted, therefore allowing for detection of rotation direction. A, B, or a combination of both are used for generating timing triggers. The index pulse allows for starting the integration process at a known angular position (index position).

In practice, the external triggering signals may be distorted, which may lead to improper operation of the integrator. One problem is associated with vibrations of the mechanical parts attached to encoders, which may result in detecting multiple closely coupled triggers (see Fig.4). To eliminate such problems, the timing module ignores any trigger occurring too close to the previous trigger. The timing module also counts the number of triggers in each full revolution, index-to-index, and compares with the expected number.

A second problem has to do with spikes produced by noise in angular encoder signals that can be detected as “false” triggers. Multiple probing is used to eliminate these disturbances. In essence, whenever a rising edge is detected on a trigger line, the inter-trigger time is saved to a temporary register (in units of 40 MHz clock ticks) and a small debouncing delay is started. If the trigger line still reads as logic high by the end of this delay, then the inter-trigger time is validated and is placed on the queue. Counting then restarts using this debouncing delay period length as a starting point instead of zero, since this delay

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV033

period has elapsed after what would have been the reset-to-zero condition for the time counter. Otherwise, if the trigger line returns to a logic low state, then the contents of the temporary register are discarded and time-counting continues to increment.

With this method, false triggers can be ignored without introducing any phase drift to the inter-trigger times. This debouncing delay is user-settable and is determined through empirical observation since noise conditions at different test stands can be unique.

EDI INTERFACE

The EDI is controlled via an Ethernet interface. Commands and data are sent through the same multichannel socket connection with separate channels devoted to streaming data and exchanging control messages. This basic functionality can be extended by adding additional channels devoted to transmission of auxiliary data and their control information.

On the client side, applications use the provided set of commands to communicate with the EDI. A subset of commands that captures the core functionality of the EDI includes:

- CONFIG: a command that sends a cluster of settings of various EDI properties and parameters,
- RUN: a request to start streaming of integrated data from all configured ADC channels,
- ABORT: a command used to stop the streaming of data (typically, the EDI is configured to send only a discrete number of integrated samples),
- TEMPERATURE: a command that reads the temperatures of all ADC modules, which can be used to correct results or check operating conditions,
- ACCELEROMETER: as an example of specialized commands added to tailor EDI functionality to a specific application, this is a request to read auxiliary analog inputs connected to an accelerometer

Depending on the configuration selected, the EDI may either perform integration using a discrete number of triggers, or it may run in a continuous mode. In this continuous mode, the EDI will not stop streaming integrated data until an abort command is received. In either mode, data is never saved to disk on the EDI but is instead always sent over the network, allowing for potential further processing of the data by other consumers on the network.

PERFORMANCE

The are some performance limitations inherent to the organization of the EDI. The integration time is provided with a resolution limited by the FPGA clock. Also, over a longer period of time, the timeline calculations will introduce additional errors due to their limited computational precision. Therefore, over a very long continuous integration measurement run (measured in hours), operations need to be periodically restarted to eliminate accumulation of errors and avoid a substantial phase shift.

Having two separate clocks in the system, with one used by the ADCs and another in the FPGA for inter-trigger timing, could also adversely impact the performance of the EDI. To address this very problem, both clocks have been synchronized.

The performance of the EDI is also strongly dependent on the performance of the ADCs, their time and temperature stability, offset, noise level, and dynamic range, among other factors.

EXTENSIBILITY

The EDI device has a configurable number of ADC integration channels with a minimum of 4 channels and has been tested with 8, 12 and 16-channel configurations. Access to additional digital and analog I/O modules allows for the instrument to also serve some auxiliary functions.

One example is the extension to read accelerometers developed for the rotating coil system for the HL-LHC AUP project [15]. The available auxiliary analog input channels are read via the FPGA and data are sent to the consumer via a separate communication channel.

Another extension developed for the same measurement system allows for receiving a stream of real-time readouts of magnet current obtained from a DCCT and distributed in the measurement hall via a fiber optic connection. This implementation uses the auxiliary digital inputs to create a receiver of encoded data packets in the FPGA.

As a future extension, the authors plan to include drift compensation and discrete Fourier transform calculations in the FPGA, therefore allowing for the conversion of integrated signals to a frequency spectrum.

SUMMARY

Building instruments from COTS parts is an attractive alternative to buying devices that don't necessarily fulfill all the requirements or building complex instruments in-house. It offers shorter development cycles, lowers development costs, and produces a device fully suited to the task.

The Extensible Digital Integrator (EDI) is an example of successfully combining several COTS hardware component and, by adding software and firmware, creating a powerful and specialized test instrument.

The EDI employs an FPGA mainly as a coprocessor, but also for synchronizing acquired analog data with external timing signals. It can be further extended to acquire and process other signals via the auxiliary I/O on the FPGA.

The use of an FPGA not only as a signal preprocessor, but as a coprocessor, can also be applied in other situations in real-time systems where data needs to be processed with hard real-time constraints.

The described digital integrator is fully functional and the conducted performance tests and comparisons with other available integrators show its satisfactory performance. In a multichannel configuration of 16 ADC channels it is capable of continuously processing 4096 triggers per second.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] B. Brown *et al.*, "A V/f-based waveform recording system for magnetic measurements", DESY-HERA Report 87-23, 1987.

[2] P. Galbraith, "Portable digital integrator", CERN Internal Technical Note 93-50, AT-MA/PF/fm, Jan. 1993, http://cdsweb.cern.ch/record/1362025/files/Portable_Digital_Integrator_1993.pdf

[3] P. Arpaia *et al.*, "A fast digital integrator for magnetic field measurements at CERN", in *2006 IEEE Instrumentation and Measurement Technology Conference Proceedings*, 2006, pp. 67-71. doi:10.1109/IMTC.2006.328175

[4] P. Arpaia, A. Masi, and G. Spiezia, "A digital integrator for fast and accurate measurement of magnetic flux by rotating coils", in *IEEE Transactions on Instrumentation and Measurement*, vol. 56, no. 2, Apr. 2007, pp. 216-220. doi:10.1109/TIM.2007.890787

[5] P. Arpaia, V. Inglese, and G. Spiezia, "Metrological characterization of an improved DSP-based on-line integrator for magnetic measurements at CERN", in *2007 Instrumentation and Measurement Technology Conference Proceedings*, 2007, pp. 1-5. doi:10.1109/IMTC.2007.379251

[6] P. Arpaia, L. Bottura, L. Fiscarelli, and L. Walckiers, "Performance of a fast digital integrator in on-field magnetic measurements for particle accelerators", in *Review of Scientific Instruments*, vol. 83, no. 2, Feb. 2012, paper 024702. doi:10.1063/1.3673000

[7] *Fast Digital Integrator FDI2056 User's Manual*, Metrolab, May 2014, https://www.metrolab.com/wp-content/uploads/2015/07/FDI2056_manual_v20_rev12.pdf

[8] G. V. Velev *et al.*, "A fast continuous magnetic field measurement system based on digital signal processors", *IEEE Trans. Appl. Superconductivity*, vol. 16, no. 2, Jun. 2006, pp. 1374-1377. doi:10.1109/TASC.2005.869702

[9] P. Arpaia, U. Cesaro, and P. Cimmino, "Enhanced fast digital integrator for magnetic measurements", *Review of Scientific Instruments*, vol. 88, no. 8, Aug. 2017, paper 085103. doi:10.1063/1.4996539

[10] X. Wang *et al.*, "A hybrid data acquisition system for magnetic measurements of accelerator magnets", Lawrence Berkeley National Laboratory, 2012, https://escholarship.org/uc/item/4gh4w163

[11] 7 Series Product Selection Guide, 2014, https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf

[12] Synchronization Explained, Jun. 2021, https://www.ni.com/en-us/support/documentation/supplemental/10/synchronization-explained.html

[13] NI PXIe-7856 Specifications, Dec. 2016, https://www.ni.com/pdf/manuals/378000b_02.pdf

[14] NI PXIe-4464 Specifications, Dec. 2016, https://www.ni.com/pdf/manuals/376165b.pdf

[15] J. M. Nogiec *et al.*, "Designing a Magnetic Measurement Data Acquisition and Control System with Reuse in Mind: A Rotating Coil System Example", submitted to the 27th International Conference on Magnet Technology, Fukuoka, Japan, Nov. 2021.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV034

EQUIPMENT AND PERSONAL PROTECTION SYSTEMS FOR THE SIRIUS BEAMLINES *

L.C. Arruda†, F.H. Cardoso, G.L.M.P. Rodrigues, G.T. Barreto, H.F. Canova, J.V.B. Franca, L.U. Camacho, M.P. Calcanha, Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil
F.A.B. Neto, F.N. Moura, Centro Nacional de Pesquisa em Energia e Materiais (CNPEM), Campinas, Brazil

Abstract

The beamlines and front ends at Sirius, the Brazilian 4th generation synchrotron light source, require monitoring and protection systems for personal and equipment safety in general, due to the high beam power dissipated along the beamline, vacuum safety, secure radiation levels, use of robots, special gases, cryogenic systems, and other highly sensitive and costly equipment throughout the facility. Two distinct programable logic controllers (PLC) were then deployed to create the Equipment Protection System (EPS) and the Personal Protection System (PPS). This work presents an overview of the EPS/PPS - requirements, architecture, design and deployment details, and commissioning results for the first set of beamlines.

INTRODUCTION

The personal protection system (PPS) and equipment protection systems (EPS) are individual per beamline and are implemented in general by two programmable logic controllers (PLCs). The PPS central process unit (CPU) is the Siemens's safety model 1516F-3 and the EPS CPU is Siemens's standard model 1516-3. Both systems use distributed I/Os with Profinet communication with the CPU. The main Graphical User Interface (GUI) is implemented by a Human Machine Interface (HMI) common to other PLC based subsystems and the Input/Output Controller (IOC) [1] is used to allow communication via OPC Unified Architecture (OPC UA) [2, 3] between the PLCs and the Experimental and Industrial Control System (EPICS) [1].

This article is divided into three parts: The first part is about the EPS, the second part is about the PPS and the third is about common subjects related to EPS and PPS.

EPS

Basic Principles

The principles of EPS are low response time, use of positive logic, distributed I/O modules, and simplified detection of the protection logic triggers. The protection logics are very similar among the beamlines, the main protection logics are related to vacuum, temperature, position, and power loss. The program is modularized by hutches, the interface data and functions with the sensors and components form an object's library. The interface with other systems happens through galvanic isolated signals, then the EPS logic and the other system logic actuate together.

There are three checklists for commissioning the EPS: I/Os signals validation, HMI validation, and protection logic validation.

Protection Logics

The protection logic related to vacuum, temperature, position, and blackout fail are following. These protection logics are triggered by an EPS reading or an interface signal with other systems.

Vacuum protection. The vacuum protection system is divided into fast vacuum protection, slow vacuum protection, and low vacuum protection.

Fast vacuum protection is treated individually by the following VAT's devices: Controller VF-2, cold cathode gauge and shutter valve [4]. The shutter valve is installed in the front-end (FE) and the gauge is installed in the vacuum path between FE and the first optical hutch. These devices aim to protect the storage ring (SR) from a high-speed shock wave that could come from the beamline. A detailed description of tests and validation of this matter can be consulted in the article [5].

Slow vacuum protection is used in ultra-high vacuum (UHV) regions, it consists in isolate vacuum paths and interlocks the valve opening in case of high pressure detected. Ionic pump and cold cathode gauge controllers diagnose high pressure through digital signals. Intended to keep the beam on SR, the FE's protection system is triggered if the ionic pump and the cold cathode gauge detect high pressure. Disconnecting the cold cathode gauge connector, it reads a very low pressure, which could indicate a safe condition to FE's protection system even if the ionic pump is turned off. To avoid this unsafe condition, the gate valves opening is allowed only if all ionic pumps before FE's shutter are detecting pressure below the limit, and all cables are connected.

Low vacuum regions are usually situated in experimental stations, with pressures between 1000 *mBar* and $1x10^{-7}$ *mBar*. The protection logics are specific for each region.

From FE's shutter, if a high pressure is detected on a vacuum path by any sensor, the protection logic is triggered closing and interlock the opening of all the gate valves in UHV downstream to the shutter where the trigger occurred. The valves opening logic between shutters are sequential, from downstream shutter to upstream shutter.

Some vacuum paths are specified as critical paths. On these critical paths, due to a fast actuating needed by the

* Work supported by the Brazilian Ministry of Science, Technology and Innovation
† lucas.arruda@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV034

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

valves, valve's close commands and shutters' close commands are sent simultaneously. The critical paths are usually presented in vacuum paths close to experimental stations' windows.

High temperature protection. The protection against high temperature is made through temperature measurement right on device structure and water flow measurement. The most used temperature sensors are PT100, PT1000, and thermocouple type K. For PT100 and PT1000, it's preferentially connected using the four wires connection. The water flow sensors are manufactured by SMC [6], model series PF3W, and uses the Kármán vortex measurement method. If the temperature read by EPS is higher than the limit or water flow is lower than the limit, the upstream shutter is closed and interlocked.

Position protection. This protection system is present usually on beam visualization devices, monochromators, and devices next to robots' workstations. The position protection ensures that the beam does not focus on parts not prepared for beam. On other devices, it also guarantees that collisions do not happen. The most used sensors are inductive or optical barrier types. In case some sensor detects an unsafe position, the upstream shutter is closed and interlocked, or the unsafe device has its motion interrupted and interlocked.

Power outage protection. This protection logic is triggered in case of fail on EPS' power supply 24 VDC. The power supply consists of the use beamline's UPS circuit, two voltage sources 24 VDC, a redundancy module, a buffer, fuses, and selectivity modules. The following Fig. 1 exposes a connection diagram of these devices.

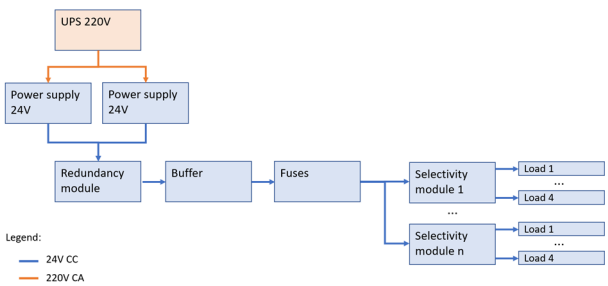


Figure 1: EPS and PPS's 24 VDC connection diagram.

The power supply failure occurs due to a beamline UPS device fail, 24 VDC power supply distribution device fail or by pressing the electricity emergency button. In case of pressed electricity emergency button, because of safe conditions defined together with Sirius' safety group, all the components of the beamline are shutdown, including ionic pumps, turbopumps, and mechanical pumps. This protection logic's priority is to keep beamline vacuum paths in a safe condition before EPS shutdown, this safe condition is guaranteed closing all the valves.

A power outage failure diagnostic is made through state signals available by the voltage sources, redundancy modules, buffers, and selectivity modules. The buffer generates a delay on the shutdown between 200 ms and 10 s, which is enough time to execute the protection logic.

Software Structure

EPS' program was separated into: Protection logic, devices' objects, other systems interface, and communication. Figure 2 represents the software structure scheme.

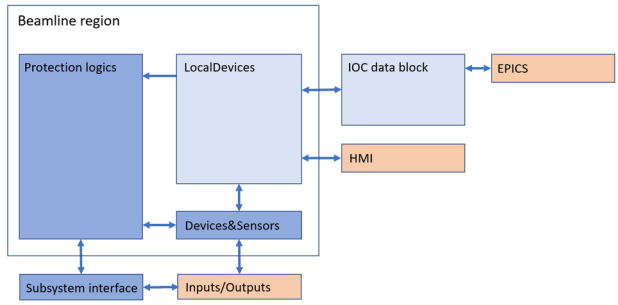


Figure 2: Scheme containing function blocks (dark blue), data blocks (light blue) and external interfaces (orange).

Objects. The objects in EPS are a set of variables and functions that allow actuating on devices, generate alarms, set settings, and collect information. As the connected signals are like each other in all beamlines, the set of object types, like gate valves, digital sensors, analog sensors, and among others, creates a library. In EPS, the objects' variables are contained in a data block called LocalDevices, and the functions of each device are contained in a function block called Devices&Sensors.

The creation of objects and the use of the library favors standardization and decreases developing time because it provides a simpler programming interface, the diagnostics variables that are used in protection logics, alarms, and devices' data accessed through HMI and EPICS communication.

By using this patterned structure of the devices, was developed a script that generates the files of IOC's settings, and the code in SCL programming language that moves LocalDevices' data blocks to EPICS' interface communication data block.

Interlock. In the functions' interface inside Devices&Sensors, there are inputs dedicated to interlock. These inputs work with positive logic and the interlock function is executed independently the others inputs' state.

Vacuum protection logic. The slow vacuum protection logic owns a patterned structure. The first part of the code generates diagnostic flags that indicates if the regions before or after the shutter are with a good vacuum. For that, each vacuum path has a logic AND or OR among the vacuum gauges depending on the path's location. Posteriorly, the generated flags of the regions before or after the shutter are used below in each valve's logic, which could trigger the device's interlocks inputs.

Commissioning

To execute the beamlines EPS' commissioning three tests was developed: signals checklist, protection logics tests, and HMI tests. The signals checklist verifies if the signal's tag on the PLC matches physically to the device and if the signal behaves as specified in the tag's

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV034

description. The protection logic tests consist in emulating in PLC all fail conditions in the beamline that triggers one of the protection logics and watch the resulting signals and components' states. In HMI all the visual indicators and commands are verified.

After these three tests validation, the hutch can operate from the equipment protection system point of view. The software division mentioned in Software structure by hutches allowed a faster availability of hutches to commissioning with the beam.

PPS

Basic Principles

The PPS consists of an automation system designed to allow facilities to operate at the appropriate security levels, determined by personal risk analysis. The solutions applied are based on engineering good practices, Brazilian and international technical standards, acquired experiences over the years in the first Brazilian synchrotron light source and from the other countries synchrotron light sources, and using certificated security components commercially available.

Protection Logics

From risk analysis in the beamlines and previous experience, four main protection logics are highlighted.

Search. The search procedure is monitored by PPS, performed by one person, and consists in a visual inspection of the hutch or a beamline's region and pressing the search buttons, to guarantee that there is nobody inside the risk region.

The search procedure must be executed by a trained and enabled person. Beyond these administrative requirements, is in development an enabling system of the search that uses an RFID card. This system will be responsible for generating a data logging containing data about the performer of the search: The date, time, and local that the search has been done. Besides, the system can set an expiration time to each RFID card.

As an improvement, light curtains were installed on the region of the hutch's door and are used to verify if more than one person goes into the hutch during the procedure. In case of interruption on the curtain's beam before all the buttons being pressed, the search process is interrupted and must be restarted. After the procedure's finalization, the light curtain inside the hutch keeps monitoring and is used as an additional emergency signal, in this case, the shutter is closed, the beamline goes to fail state and the doors are unlocked.

Security locks. On the hutches' doors were installed security locks from different manufactures to offer diversity and redundancy. One of the locks counts with an electromagnetic lock and the other with a mechanical lock, so that exists a position monitoring and a lock that generates a force maintaining the door in a safe position.

Enabling keys. Each beamline count with a set of keys monitored by PPS. Being one for each hutch, a general enabling key used by the radiological protection group, and a key for reset. The chicanes own monitored keys by PPS and use the trapped key system.

Emergency buttons. The types of emergency buttons were reduced to only two buttons, being the personal protection emergency button and the electrical emergency button. The personal protection buttons are distributed over the beamline and monitored by PPS to react in case of pressing, immediately closing the shutters and disabling other systems that could exist in the beamlines, like robots.

Software Structure

The PPS's program is separated into some logical parts: safety protection logic, non-safety protection logic, data blocks to create an interface between safety and non-safety logic, and data blocks used for communication. Figure 3 represents the software structure scheme.

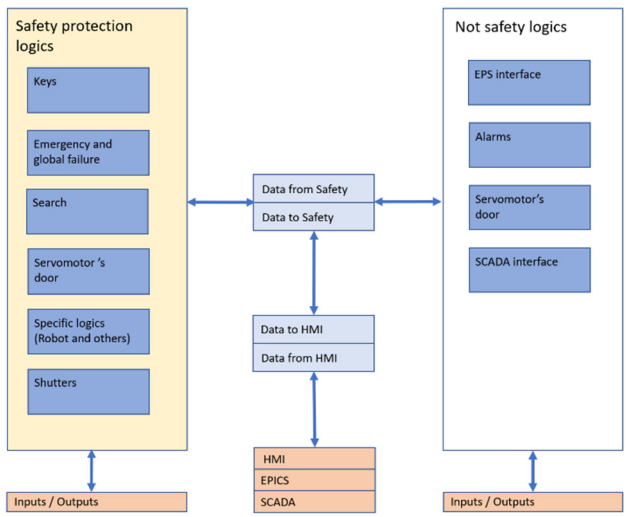


Figure 3: Scheme containing the main function blocks (dark blue), data blocks (light blue) and external interfaces (orange) to the PPS.

The search and shutter function blocks are equal functions for all the regions that need these functionalities.

The HMI, EPICS, and SCADA communication data blocks were separated from the safety data for security.

Initial Tests of the Protection Logics

To validate the PPS's logic before using the program in the beamline were developed a simulation software by tables using a virtual controller through an API from S7-PLCSIM Advanced and an application based on SimTableApplication, made available by Siemens [7].

The program inputs consist of tables with initial beamline conditions, inputs with changed values in an interval of time, and the expected states to the outputs. By executing the program, the generated outputs by the virtual controller are compared with the expected states and are generated a report indicating the comparison result.

Commissioning

To execute the beamlines PPS's commissioning, initial tests of hardware operation are made (indicator lights, buttons, safety locks, and others), following is a checklist

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV034

simulating in the beamline all the conditions of each protection logic and possible human fails during the safety procedures. This inspection is made before the beginning of the beamline operation and is repeated periodically by the radiological protection group.

COMMON TOPICS

Interface with MPS

The interface between the SR machine protection system (MPS) and PPS is made to guarantee the FE Gamma shutter closed during the injection and to provide a redundancy in case of gamma shutter closing failure.

The interface between MPS and EPS is made to verify if there is beam in FE, provide, and receive vacuum status and allow or not beam in the beamline.

Although is necessary the exchange of signals between the beamline's protection and MPS, the beam availability on SR was considered and preserved on the maximum to a beamline fail does not affect all the other beamlines through the SR.

Interface with EPICS

To allow the integration of the EPS and PPS with EPICS to use Epics Process Variables (PVs) in control interfaces and Archiver [8], an IOC was created.

The EPICS is a standard tool used in beamline control systems and that's the reason to make the EPS and PPS data available in EPICS.

In EPS, the PVs are used principally to actuate a device remotely and inform the diagnostic related to interlock logics, limits, and status flags with or without reset from the read values and analyzed by the sensors and components. In PPS, the PVs are mainly used to inform the status and actuate the shutters, inform the radiological protection keys status, emergency or fail in the beamline and give commands to reset.

Human Machine Interface

The HMIs are installed in the beamlines' user's room and provides the PLC based systems GUIs. A more high-level view of the Sirius' facilities is shown in the SCADA system in development [9]. Other screens can be developed in EPICS by other teams using the available PVs.

EPS and PPS. The HMIs' screens of the beamline are divided per subsystem; therefore, PPS and EPS screens are separated. In EPS and PPS screens the information is divided in an initial screen by hutch of the beamline. The screen consists of an installation map, the main signals of diagnostic and operation. Some examples of EPS and PPS screens are shown in Fig. 4 and Fig. 5 following.

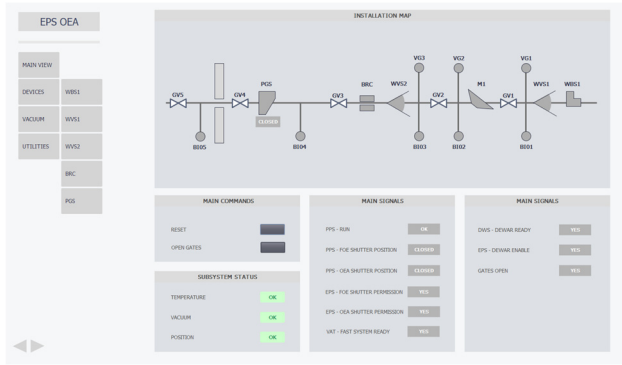


Figure 4: The main screen of EPS HMI of a hutch from a beamline representing a P&ID simplified diagram with components status, protection logics status, reset button, and the main interface signals between EPS and PPS.

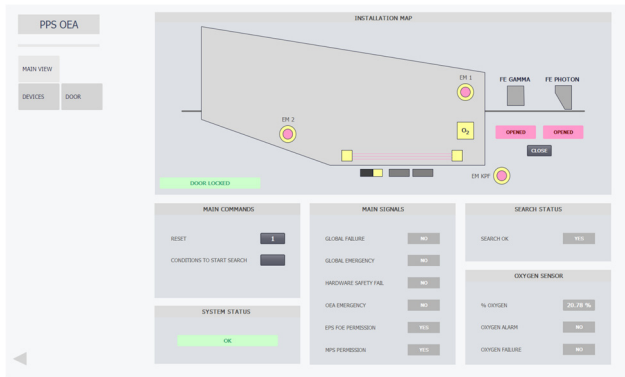


Figure 5: The main screen of PPS HMI showing FE's shutter status and components status inside the hutch, main diagnostic internal signals, interface signals with EPS, reset button, oxygen, and search status.

Alarms. An alarms table in Fig. 6 concentrates EPS and PPS alarms. The different types of alarms are encoded by colours: Red indicates that the alarm is active, and the beamline cannot operate, that is, the beam is no longer enabled to get to the sample. Yellow indicates a warning, that is a condition that requires attention, but does not prevent the beamline operation. Green indicates that fail or warning was active, but the problem is solved.

This table has two types of visualization called normal and buffer. On normal mode, the status of the alarm's status is shown in real-time, and on buffer mode all the historic of the alarm activation, deactivation, and acknowledgment are present. The buffer data can be cleared by using the button "clear" on the screen, this button has restricted access by password.

To make a more efficient visualization a filter was created that allows filtering the alarms shown in the table per hutch and subsystem. This filter uses as input the alarm description text that owns at beginning of the text the subsystem and the origin hutch.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV034

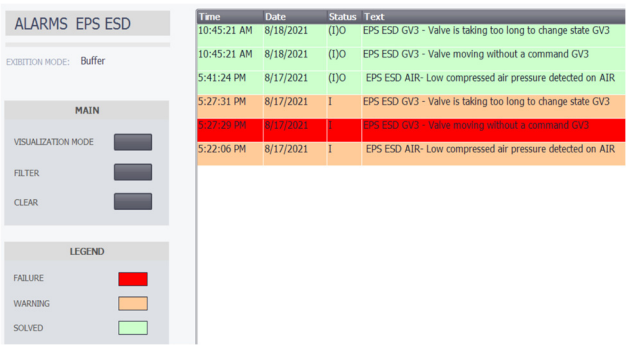


Figure 6: Sequence of alarms registered when there was a lack of compressed air on D hutch from beamline Carnauba.

For example, analyzing the alarms in Fig. 6 is possible to conclude that probably the lack of air made the valves move to a position different from that EPS was commanding, it caused a failure on the beamline, and that the insufficient pressure did not close the valve completely. It's also observed when the pressure was back to normal and when the valve moved to a close position. Only using Archiver [8], this analysis of causes and effects could consume a time considerably higher.

Sirius Improvements and UVX

In UVX, EPS and PPS worked differently than was projected to Sirius. The principal characteristics in UVX were: There was only one CPU per beamline from manufacturer Pepperl Fuchs [10] for both systems EPS and PPS, personal safety logics were executed using safety relays, distributed I/Os with ASI communication, and the interface with the user was made through panels with pushbuttons and a graphic interface developed by LNLS called Superinter to generation and verification of alarms communicated by RS232. In this system, no analog sensors were reading, only digitals because of the used hardware characteristics.

The main improvements in Sirius are related to infrastructure, software structure, communication, user interfaces, EPICS communication, and alarms diagnostics.

ACKNOWLEDGMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, and Innovation and the contribution of the LNLS team involved in the development, installation, and commissioning in the EPS and PPS subsystems.

REFERENCES

[1] Experimental Physics and Industrial Control System, https://epics.anl.gov/

[2] OPC Foundation, https://opcfoundation.org/about/opc-technologies/opc-ua/

[3] opcUaUnifiedAutomation, https://github.com/bkuner/opcUaUnifiedAutomation

[4] VAT, https://www.vatvalve.com/

[5] Oliveira, H.G.P. *et al.*, "Validation results for Sirius APU19 Front end prototype", in *Proc. MEDSI 2018*, Paris, France, July 2018. doi:10.18429/JACoW-MEDSI2018-WEPH39

[6] SMC PNEUMATICS, https://www.smcpneumatics.com/

[7] Getting started with S7-PLCSIM Advanced and simulation tables, https://support.industry.siemens.com/cs/document/109759047/getting-started-with-s7-plcsim-advanced-and-simulation-tables?dti=0&lc=en-NO

[8] Experimental Physics and Industrial Control System, https://epics.anl.gov/extensions/ar/index.php

[9] Arruda, L.C. *et al.*, "Supervisory System for the Sirius Scientific Facilities", presented at ICALEPCS 2021, Shanghai, China, Oct. 2021, paper THPV001, this conference.

[10] Pepperl Fuchs, https://www.pepperl-fuchs.com/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV036

THE LMJ TARGET CHAMBER DIAGNOSTIC MODULE

R. Clot, CEA, Le Barp, France

Abstract

The Laser MegaJoule (LMJ), the French 176-beam laser facility, is located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy on targets, for high energy density physics experiments, including fusion experiments. The first bundle of 8-beams was commissioned in October 2014. By the end of 2021, ten bundles of 8-beams are expected to be fully operational.

Due to the energy levels achieved, the optical components located at the end of the bundles are highly subject to damage stresses. This is particularly the case with vacuum windows whose integrity is critical.

To measure these damages, identify the growth laws, and prevent their degradation (through blockers), the Target Chamber Diagnostic Module (called by its french acronym MDCC) was integrated into the LMJ installation in 2019. This diagnostic, which also measures the windows transmission rate, as well as the spatial energy distribution at the end of the bundles, has been designed to operate automatically at night, between two experiments.

This presentation describes this three years feedback of MDCC. It also presents the areas for improvement which have been identified to optimize its efficiency and reduce its timeline.

INTRODUCTION

The laser Megajoule (LMJ) facility, developed by the "Commissariat à l'Energie Atomique et aux Energies Alternatives" (CEA), is designed to provide the experimental capabilities to study High Energy Density Physics (HEDP). The LMJ is a keystone of the Simulation Program, which combines improvement of physics models, high performance numerical simulation, and experimental validation, in order to guarantee the safety and the reliability of French deterrent weapons. Once fully operationnal, the LMJ will deliver a total energy of 1.4 MJ of 0.35 µm (3ω) light and a maximum power of 400 TW.

The LMJ is sized to accommodate 176 beams grouped into 22 bundles of 8 beams. These will be located in the four laser bays arranged on both sides of the central target bay of 60-meter diameter and 40-meter height. The target chamber and the associated equipment are located in the center of the target bay.

Due to the energy levels achieved, the optical components located at the end of the bundles are highly subject to damage stresses. This is particularly the case with vacuum windows whose integrity is critical as it is a safety component which constitutes the vacuum limit of the chamber where the experiments take place. It's also an expensive component that we need to take care of.

To measure the damages they suffer, identify the growth laws, and prevent their degradation, the Target Chamber Diagnostic Module (called by its French acronym MDCC) was integrated into the LMJ facility at the end of 2019. This diagnostic inqtrument, which also measures the windows transmission rate, as well as the spatial energy distribution at the end of the bundles, has been designed to operate automatically by night, between two experiments.

This paper reminds the LMJ facility principle before presenting the MDCC and its functionalities with a focus on the measurement of optical component damaging. It also presents the feedback of the measurement sequence timeline and the work achieved to reduce it and make it fit in the facility timeline.

LMJ OPERATING REMINDER

The LMJ 176 beams (37 x 35.6 cm² each) are grouped into 22 bundles of 8 beams. In the switchyards, each individual bundle is divided into two quads of 4 beams, the basic independent unit for experiments, which are directed to the upper and lower hemispheres of the target chamber.

Basically, an LMJ laser beam line is composed of three parts: the front-end, the amplifying section, the switchyard and the final optics assembly.

The front end delivers the initial laser pulse (up to 500mJ). It provides the desired temporal pulse shape and spatial energy profile.

The initial pulse leaving the front end is amplified four times through two amplifiers, in order to obtain the energy required for the experiments (up to 15 kJ at 1ω per beam). Positioned between the two amplifiers, focusing lenses, associated to a diaphragm (spatial filter pinhole), take out the parasitic (noise) beams that may arise. Beyond the two amplifiers is a reflecting mirror (M1), making the four passes possible through angular multiplexing, as shown on Fig. 1. The surface of this mirror is deformable (being controlled by electro-mechanical actuators), allowing beam wavefront distortions to be controlled.

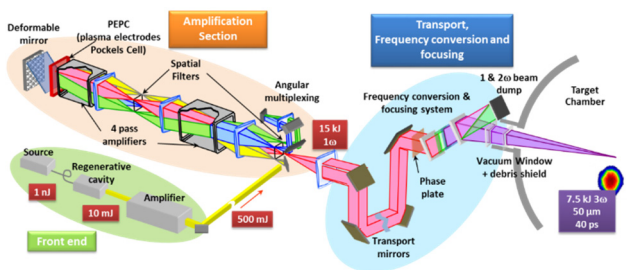


Figure 1: Laser beamline schematic diagram.

The 8 beams coming from the amplification section are divided into two quads. Each quad is transported over more than 40 meters into the target bay and is directed to the upper or the lower hemisphere of the target chamber using six transport mirrors per beam. Each quad arrives into a frequency conversion and focusing system (SCF). Inside the SCF, the beams frequency is changed from infrared (1,053

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV036

nm) to ultraviolet (351 nm). The frequency conversion is realized by two KDP crystals, with a global efficiency of about 60%. The beams are focused on target using a 3ω focusing grating. This specific component allows spectral separation (to ensure only 351 nm wavelength reaches the target).

MDCC MEASUREMENTS

The MDCC achieves three different measurements on LMJ facility.

The first one is the damaging of the end of bundle optical components: the vacuum windows and the 3ω gratings. The vacuum windows being the most important optical component, they make the damaging measurement critical for LMJ facility security.

The second measurement is the vacuum windows transmission rate. It consists in an intensity comparison of a monitored ultraviolet LED signal send through the vacuum window, and the returned signal (reflected) measured by the MDCC.

The third measurement is the energy distribution at the end of the bundles, which is measured by the MDCC on the vacuum window on a 1J shot to the target chamber.

MDCC PRESENTATION

The MDCC is made up of two subsystems: an Optical Block and a Motion Block (Fig. 2).

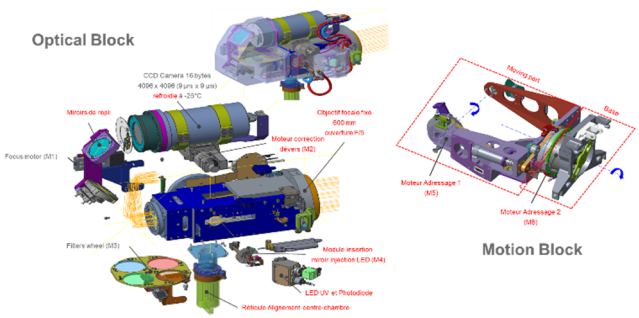


Figure 2: MDCC presentation.

The Optical Block is assigned to the acquisition part of measurement. It's especially made up of a large lens and a CCD camera which receive and acquire the signals. Transmission between them is handled by two redirection mirrors which enable the camera focus thanks to a motor. Three other motors are used on the Optical Block to correct the camera camber, to move a wheel to filter wavelengths, and to insert a mirror dedicated to a UV LED redirection for the transmission rate measurement.

The Motion Block is assigned to the diagnostic positioning. It carries the Optical Block and enables to address the various optical components of the installation thanks to two motors and two rotation angles.

DAMAGING MEASUREMENT PROCESS

The acquisition of damages on vacuum windows is realized shot to shot at night with the MDCC while teams are resting (Fig. 3).

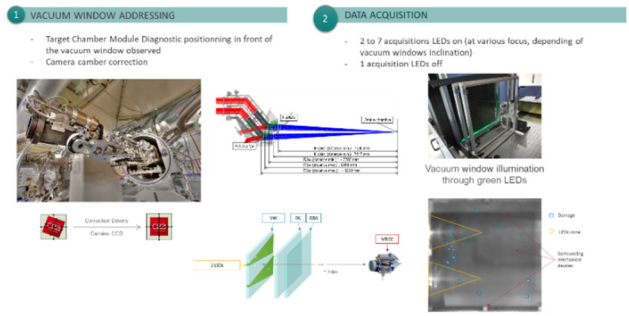


Figure 3: Damaging measurement process.

In the first part of the measurement, the MDCC positions itself in front of the vacuum window observed, 7 to 8 meters away. The camera camber, dependent of the component tracked, is corrected at the same time to center the window on the camera.

Two green LEDs on the side of the vacuum window are used to light the surface and optimize the damage tracking. Each damage becomes a bright spot visible on acquisitions. The illustration down center shows the LEDs cone in yellow and the damages surrounded by blue squares.

Several acquisitions are made at various camera focus to take in account the different vacuum window inclination and the camera depth of field.

A final acquisition, with LEDs off, is made to substract a noise image to the useful ones and also enhance acquisition analysis.

DAMAGING MEASUREMENT STAKES

The vacuum window damaging measurement is the most critical and has 3 determining stakes for LMJ facility (Fig. 4).

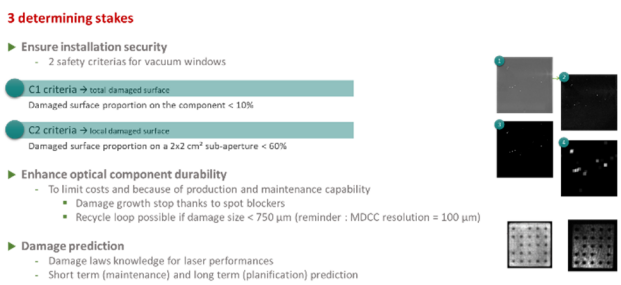


Figure 4: Damaging measurements stakes.

Thanks to MDCC, we're able to detect damages of a 50 micrometers size. This size corresponds to the triggering of damages and the beginning of their growth. To characterize them we track 2 safety criteria thanks to a software analysis with one being the most restrictive. By keeping C2 criteria under 60% for any 2x2cm² sub-aperture, we ensure the LMJ installation security.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV036

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A second stake of the damage measurement is to enhance our optical components durability. Our goal is to stop the growth of damages from 400 µm as we are unable to repair damages bigger than 750 µm.

The challenge is therefore to be able to monitor damage growth in noisy images before damage size reaches 5 pixels.

We also have to set efficient algorithms to detect and track damages.

To prevent the growth of the damages and keep them under the limit which allows their repair, we developed a new functionality on the LMJ installation. It enables to set spot blockers on the energy spatial distribution at the start of the laser bundles, and also prevents spreading energy on the most damaged surfaces of the vacuum windows. The pictures down right show a spot blockers grid at the start and at the end of a bundle.

The third stake of the damage measurement is the prediction capability. We need to identify the damage laws to improve laser performances, optical components maintenance at short term, and installation planification at longer term.

Our strategy is also to protect our end of bundle optical components is based on a complete scheme including acquisition, damage detection, damages growth brake and recycling loop of the components.

The first steps are already in use, whereas the first recycled components will be back on the installation around 2025.

SEQUENCE TIMELINE

MDCC has been designed to operate automatically by night, without needing any operator intervention. This is very important for the LMJ installation as experiments are prepared from the early morning for the shot to happen in the evening. The damage measurement at night is also critical to give the green light to the experiment of the next day.

After MDCC integration at the end of 2018, the sequence duration put in the front that by extrapolation it was well too long and totally incompatible with installation timeline.

So we launched a research for solutions to shorten the sequence and make it fit the 4 hour time slot allocated in the targeted installation timeline.

TIMELINE IMPROVEMENTS

Thanks to an efficient team work, 12 improvements have been identified to reduce the damaging measurement sequence timeline. They have been classified in 4 themes whether they relate to the camera, the MDCC mechanics, the measurement principles or the command control.

Two of them have been dropped as they were either too complex to implement or too restrictive on performances.

The other ten have been considered in view of their gain on the timeline and our ability to implement them quickly or in the long-term (Fig. 5).

THEME	IMPROVEMENT	IMPLEMENTATION COMPLEXITY [1]	TIME SAVER FROM MDCC ORIGINAL CONFIGURATION (22h11)[2]	PERFORMANCES IMPACT
CAMERA	Upgrade of acquisition card electronics	Average	4h30	-
	Camera model change (CCD to CMOS)	High	6h	-
MECHANICALS	Compensation of vacuum window tilt through POCC	Very high	8h	Very good
	Compensation of vacuum window tilt through MDCC	High	8h	Very good
	Addressing motor speed enhancement	Very low	35mn	Minor or none
MEASUREMENT	Optimisation of useful acquisition number	Low	- 6mn	Very good
	Reduction of noise acquisitions number	Low	4h10	Minor
	Decrease of acquisitions resolution	Very low	6h	Very bad
COMMAND CONTROL	Deletion of sequence configuration bypassing	Very low	10 mn	-
	Deletion of double results production	Very low	3h20	-
	Enhancement of results production	Low	1h45	-
	Parallelization of configuration and results production	Average	3h20	-

[1] Cost/Time compromise
[2] Improvements combination reduce timesaver compared to sum of individual improvement timesavers

Figure 5: Sequence timeline improvements.

IMPROVEMENTS IMPLEMENTATION PLAN

To reduce the sequence timeline according to the LMJ installation needs, we built an implementation plan of the selected improvements. We planned each other into one of 3 phases forecasted between 2020 and 2024 (Fig. 6).

Phase 0 2020 Summer

THEME	IMPROVEMENT	IMPLEMENTATION STATE
COMMAND CONTROL	Deletion of sequence configuration bypassing	Achieved in 2020
	Deletion of double results production	Achieved in 2020

Phase 1 2021 Summer

THEME	IMPROVEMENT	IMPLEMENTATION STATE
CAMERA	Upgrade of acquisition card electronics	Achieved in 2021
MESURE	Optimisation of useful acquisition number	Achieved in 2021
	Reduction of noise acquisitions number	Achieved in 2021
COMMAND CONTROL	Enhancement of results production	Achieved in 2021

Phase 2 2024 Summer

THEME	IMPROVEMENT	IMPLEMENTATION PLANNING
CAMERA	Camera model change (CCD to CMOS)	2024-2025
MECHANICALS	Addressing motor speed enhancement	2022
	Compensation of vacuum window tilt through MDCC	2024
COMMAND CONTROL	Parallelization of configuration and results production	2024

Figure 6: Improvements implementation plan.

The first one has been achieved in 2020 and the second mid-2021. The results confirm the expectations we made as we already reduced the sequence timeline by more than 50% as foreseen.

Without surprise, the mechanicals improvements and the camera change will be the latest to be implemented, as they lead to an important work of preparation and maintenance.

EVOLUTION OF SEQUENCE TIMELINE

The following diagram (Fig. 7) presents the damaging sequence timeline evolution thanks to the implementation plan previously described.

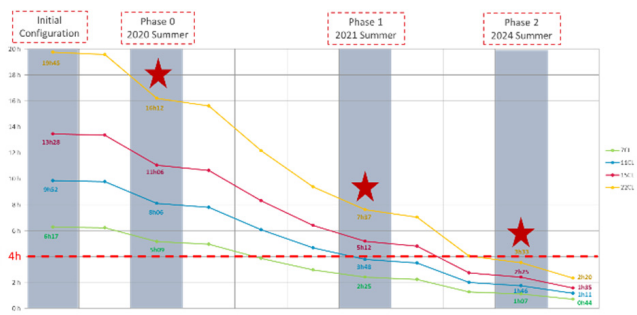


Figure 7: Sequence timeline evolution.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV036

It shows the way the sequence timeline shortens through the 3 phases displayed in the previous slide.

Each colored line matches a configuration of LMJ, that is to say the number of laser bundles used for the main milestones of the installation.

Thanks to Phase 1, we're now already able to run the damaging measurement sequence in less than 4 hours for 11 laser bundles, which is our next milestone in 2022.

Phase 2 will be a prequisite for the next ones when we will use 15 and 22 laser bundles, with an expectation of hardly 2 hours and 20 minutes for LMJ nominal configuration.

It will mean MDCC came a long way if we consider where is started in term of sequence timeline.

SUMMARY

To sum up, there are six things to remember:

1. MDCC is a recent and autonomous laser diagnostic which achieves three measurements.
2. The vacuum windows damaging one is the most critical for LMJ installation as it's the greenlight for the next experiment.
3. MDCC also enables to ensure installation security and enhance optical components durability.
4. The first sequences led to a warning on the duration of the damaging measurement sequence.
5. To reduce it we identified various improvements that we started to implement, with the main ones to come.
6. The evolution of sequence timeline is promising to fulfil the installation needs and our expectations.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV037

DEVELOPMENT OF A VOLTAGE INTERLOCK SYSTEM FOR NORMAL-CONDUCTING MAGNETS IN THE NEUTRINO EXPERIMENTAL FACILITY AT J-PARC

K. Nakayoshi*, K. Sakashita, Y. Fujii
High Energy Accelerator Research Organization (KEK), Tsukuba, Japan

Abstract

We are upgrading the neutrino experimental facility beamline at J-PARC to realize its 1.3 MW operation. One of the upgrade items is to strengthen the machine protection interlocks at the beamline. So far, we have developed an interlock system that monitors the output current of the power supplies for the normal-conducting magnets in the primary beamline. On the other hand, a coil-short in one of the bending magnets at a beam transport line (3-50BT) at J-PARC happened in 2019, and it caused a drift of the beam orbit over time. Our present interlock system cannot detect a similar coil-short in the magnet, while such a change of the beam orbit may cause serious damages. One possible way to detect such a coil-short is to monitor the voltage of the magnet coil. Actually, a significant voltage drop between layers of the coil was observed for the 3-50BT magnet coil-short. Focusing on that fact, we are developing a system that continuously monitors the voltage value of the magnets at the primary beamline and issues an interlock when there is a fluctuation exceeding a threshold value. We report on the progress of the development of this system.

BACKGROUND

At the neutrino experimental facility at J-PARC, a large amount of neutrinos are generated using a high-intensity proton beam extracted from the Main Ring Synchrotron (MR) and sent towards the Super-Kamiokande detector, 295 km away, for a long baseline neutrino oscillation experiment (the T2K experiment) [1]. The MR stopped operation in July 2021 and upgrades, such as replacing the power supplies are ongoing. The beam to the T2K experiment is scheduled to resume in the fall of 2022. Also, the neutrino beamline equipment is being upgraded to support a beam intensity enhancement [2]. Figure 1 shows the primary proton beamline of the neutrino experimental facility. The proton beam extracted from the MR is transported to the graphite target 240 m away by 14 doublet super-conducting magnets (hatched in yellow) and 21 normal-conducting (NC) magnets (hatched in blue). If an abnormality occurs in the transport system of the primary proton beamline during beam operation, the high-intensity proton beam may deviate from the normal orbit and hit the beamline equipment. In that case, the thermal shock of the high-intensity proton beam could cause serious damages to the beamline equipment, which would take a long time to recover. In order to avoid such a situation, we have strengthened the interlock so far. For NC magnets, we have developed an interlock system for detecting current fluctuations in the NC power supplies and have been operating this system from 2012 to the present [3, 4]. When an interlayer short circuit occurred in a dipole magnet coil of J-PARC's beam transport line in 2019, there was no change in the current Value of the power supply, but fluctuations in the voltage of the corresponding magnet coil were observed [5]. This means that our current fluctuation interlock cannot detect an interlayer short circuit of the NC magnets. In order to further strengthen the interlock of the NC magnet system, we are developing a system that detects a voltage fluctuation of the NC magnets and issues an interlock. The target value of the voltage fluctuation detection system was set to 1%.

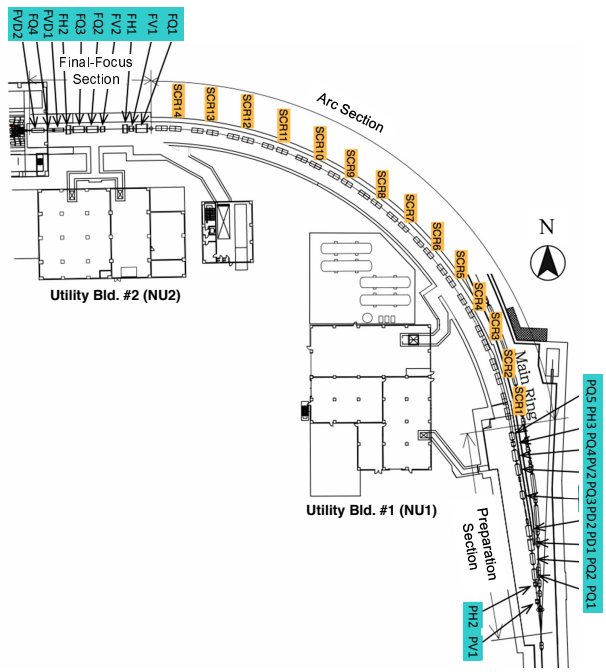


Figure 1: The primary proton beamline of the neutrino experimental facility at J-PARC.

PRELIMINARY STUDY

How Can We Measure the Magnet Coil Voltage?

Before developing the voltage interlock system, we made a preliminary measurement of the magnet coil voltage. During the accelerator operation, the voltage of the magnet coil

* kazuo.nakayoshi@kek.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV037

was measured to check its noise level, and whether an interlock that detects voltage fluctuations and issues an interlock is feasible. The measurement was performed for 5 hours using a digital multimeter Keithley 2701. Figure 2 shows a histogram of the results of the measurement. The average value of the histogram in Fig. 2 is 49.5 V and the full width is 0.120 V (*full width/average* = 0.24%). It was found that the voltage of the magnet coil could be measured at the power supply on the ground floor with the required accuracy even when the accelerator was in operation.

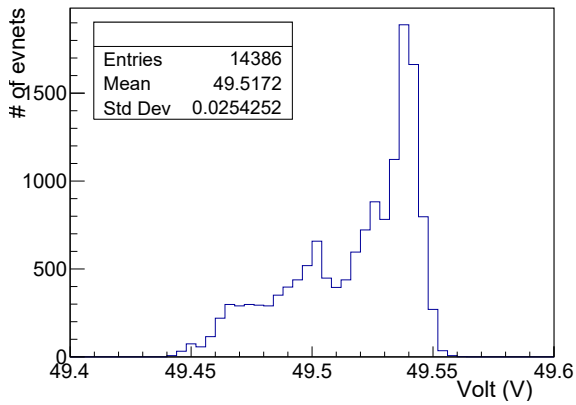


Figure 2: The result of a preliminary measurement of the magnet coil voltage.

What Kind of System We Will Adopt?

Figure 3 shows the equivalent circuit of our voltage fluctuation detection circuit. In order to minimize the magnet current deviation, we considered a voltage divier circuit. The NC magnet impedance is 39.3 mΩ. Taking the power consumption of the voltage fluctuation detection circuit into account, we decided to set the impedance R_s of the voltage divider circuit + ADC to 10 kΩ ≤ R_s ≤ 10 MΩ.

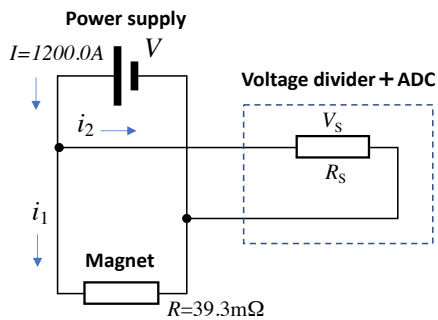


Figure 3: Equivalent circuit when the voltage divider circuit and ADC are connected to the magnet and power supply.

In order to realize a small latency and an intelligence function, we are evaluating a system using an Arduino UNO [6] left side of Fig. 4 and external ADC. For this study, we evaluated a Pmod-type ADC board [7] equipped with AD7685 (16 bit, 250 kSPS), right side of Fig. 4. The input impedance of this board is as small as 0.4 kΩ, and since it does not meet the impedance required by this system, the resistance of the board was changed to 10 kΩ. An external resistor of 190 kΩ was attached in front of the ADC to divides the voltage at 20:1.

We will use EPICS as a control framework for the voltage interlock system like other interlocks in the neutrino facility. Also, a Raspberry Pi 3 [8] is used as an interface between the EPICS IOC and Arduino.

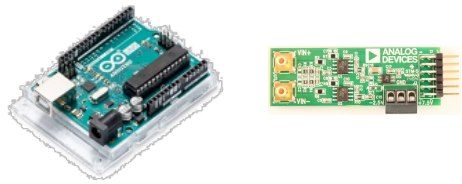


Figure 4: Arduino UNO (left) and Pmod type ADC (right).

VALIDATION OF THE USE OF ARDUINO AND ADC

Check Feasibility of ADC Readout by Arduino With SPI

Since the Pmod ADC board supports SPI (Serial Peripheral Interface, an interface bus commonly used to send data between microcontrollers and peripherals), we implemented SPI transfer with the Arduino. Figure 5 shows the wiring of the Arduino and Pmod ADC for SPI transfer. It was confirmed that the data can be read by SPI communication using a 3-wire mode with no busy indicator.

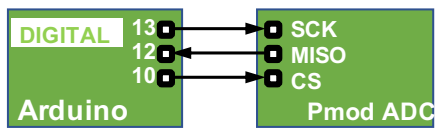


Figure 5: Wiring for SPI between the Arduino and Pmod ADC.

To check if the Arduino could read the ADC sampling data correctly, we read a sine wave output from a function generator using the Arduino. Figure 6 shows a graph of a 3 Hz, 1 V_{pp} input signal obtained by the Arduino.

We also measured the linearity of the Pmod ADC. The input to the Pmod ADC is the output of the DC power supply. Figure 7 shows the results.

Check the Functions Required for the Interlock System

We checked the Arduino for the following features needed for the interlock function.

- Checking the ability to compare read data with thresholds
- Implementation of threshold setting using interrupts
- Confirmation that the signal can be output as an interlock

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

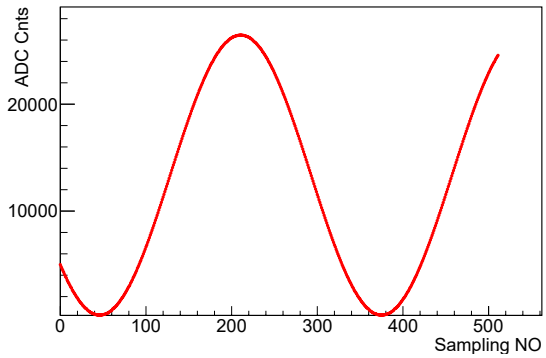


Figure 6: Sine waveform acquired by the Arduino and Pmod ADC.

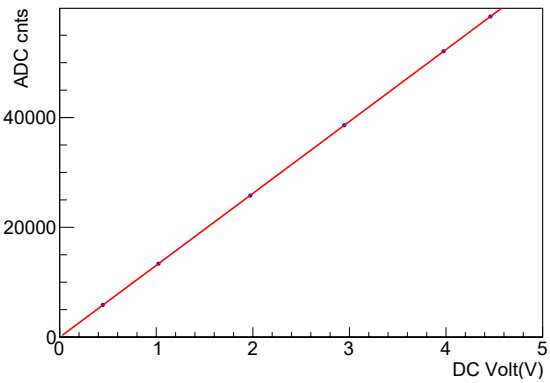


Figure 7: Results of linearity check for Pmod ADC.

The prototype provided the three operating modes shown in Fig. 8. In the normal mode, the Arduino reads the ADC

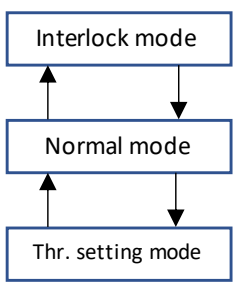


Figure 8: Operation modes for the Arduino.

value at 1 kHz, compares it with the threshold value, and transfers the ADC and threshold value to the Raspberry Pi at 1 Hz. When the ADC value exceeds the threshold value, the mode transitions to the interlock mode. The Arduino sets the digital output pin to HIGH and issues an interlock. In the threshold setting mode, the threshold value can be changed by sending a new threshold value from the Raspberry Pi to the Arduino. Changing the operation mode between the normal mode and the threshold setting mode is done utilizing the interrupt function of the Arduino.

Threshold Setting Using Interrupts

Arduino interrupts were used to set the threshold for detecting voltage fluctuations. By turning on the interrupt signal for the Arduino from the GPIO pin of Raspberry Pi, an interrupt for the Arduino is generated. The function called at the time of interrupt reads the threshold value written before the interrupt from the buffer and updates the threshold.

PROTOTYPE DEVELOPMENT

Prototype Configuration

Figure 9 shows the configuration of the prototype. The Pmod ADC board and Arduino are connected via a conversion board. The Arduino is connected to a relay board to output the interlock signal. The Arduino and the Raspberry Pi are connected with a USB cable and serial transfer is possible. Also, a cable is connected between the Raspberry Pi and Arduino for an interrupt signal. The Raspberry Pi communicates with the EPICS IOC via a media converter.

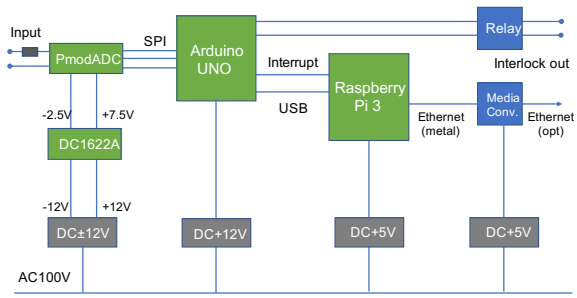


Figure 9: Diagram of the prototype.

The Prototype in a Box

The prototype has been made into a single box for future performance evaluation tests (Fig. 10). The relay module and media converter for Ethernet keep this box electrically isolated from downstream systems. The size of the box is 430 mm × 330 mm × 115 mm.

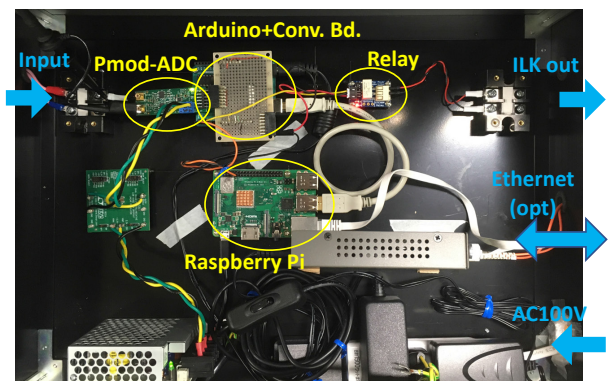


Figure 10: The prototype in a box.

PERFORMANCE EVALUATION

Long-Term Stability Test

To evaluate the performance of the prototype, a stability test was conducted for 8 hours. As an input signal, a constant DC voltage of 10 V was injected using a DC power

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV037

supply. The results are shown in Fig. 11. We found some noise spikes, but they were smaller than our target of <1% variation (red dashed line in the figure). We are now trying to determine the cause of these nose spikes and how to reduce them.

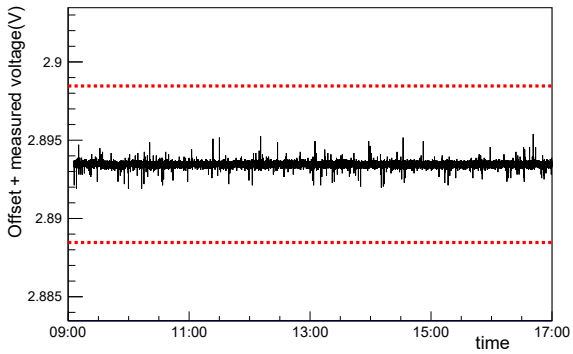


Figure 11: The result of the long-term stability test.

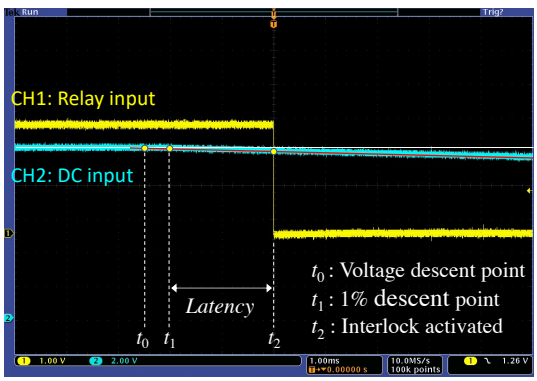


Figure 12: The result of the interlock latency measurement.

Interlock Latency Measurement

To measure the latency of the interlock, we used a DC power supply with an output of 10 V. The threshold value was set to ±1% of the average value, and the output voltage was changed to trigger the interlock by hand. The latency time was measured by an oscilloscope (Fig. 12). In Fig. 12, CH2 (blue line) is an input to the ADC from the DC power supply, and CH1(yellow line) is an output of the Arduino to the relay. t_0 is the time when the voltage starts to decrese. t_1 is the time when the voltage has decreased by 1%. t_2 is the time the interlock was activated. We fit the CH2 data with two linear functions and set their intersection point as t_0. The latency time was calculated as t_2 - t_1. Ten measurements were taken, and the average latency time was 2.1 ms.

SUMMARY AND FUTURE PLANS

In order to strengthen the interlock of the primary proton beamline NC magnets of the J-PARC neutrino experimental facility, we are developing a system that continuously monitors the voltage of the magnet coils and issues an interlock when the threshold is exceeded. We developed a prototype of the system. For the next step, the prototype will be attached to the NC power supply for testing and improvement. After that, the actual system will be fabricated, and when the experiment resumes in the fall of 2022, the beamline will be operated with this interlock installed.

REFERENCES

[1] K. Abe *et al.*, "The T2K experiment", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 659, no. 1, pp. 106-135, Dec. 2011. doi:10.1016/j.nima.2011.06.067

[2] K. Abe *et al.*, "J-PARC Neutrino Beamline Upgrade Technical Design Report", 2019. arXiv:1908.05141

[3] K. Nakayoshi, Y. Fujii, and K. Sakashita, "Improvements in the T2K Primary Beamline Control System", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, 2013, paper TUCOCA10, pp. 940-943. https://jacow.org/ICALEPCS2013/papers/tucoca10.pdf

[4] K. Nakayoshi, Y. Fujii, and K. Sakashita, "Upgrade of the Control and Interlock Systems for the Magnet Power Supplies in T2K Primary Beamline", in *Proc. ICALEPCS2015*, Melbourne, Australia, Oct. 2015, pp. 160–162. doi:10.18429/JACoW-ICALEPCS2015-MOPGF030

[5] J. Takano, "Layer Short Circuits of Bending Magnets in the J-PARC Main Ring and 3-50BT", in *Proc. J-PARC2019*, vol. 33, p. 011037, 2021. doi:10.7566/JPSCP.33.011037

[6] Arduino website: https://www.arduino.cc/

[7] Analog Devices, PulSAR ADC PMODs: https://wiki.analog.com/resources/eval/user-guides/circuits-from-the-lab/pulsar-adc-pmods

[8] Raspberry Pi Foundation website: https://www.raspberrypi.org/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV038

PERFORMANCE VERIFICATION OF NEW MACHINE PROTECTION SYSTEM PROTOTYPE FOR RIKEN RI BEAM FACTORY

M. Komiyama†, A. Uchiyama, M. Fujimaki, K. Kumagai, N. Fukunishi, RIKEN Nishina Center, Wako, Saitama, Japan
T. Nakamura, M. Hamanaka, SHI Accelerator Service, Ltd., Shinagawa, Tokyo, Japan

Abstract

We herein report on the performance verification of a new machine protection system prototype for the RIKEN Radioactive Isotope Beam Factory (RIBF) and on the study to improve its performance. This prototype has been developed to update the existing beam interlock system (BIS) that has been in operation since 2006. The new system, as was the BIS, was configured using programmable logic controllers (PLC).

We applied the prototype to a small part of RIBF and started its operation in September 2020. It consists of two separate PLC stations, has 28 digital and 23 analog inputs as interlock signals, and has five digital outputs used to stop a beam. The observed response time averaged 2 ms and 5.4 ms, within one station and with both stations, respectively. When deploying the prototype at the same scale as the BIS, which consists of five PLC stations with roughly 400 signals, the resulting performance would barely meet our requirements. Further, there is a risk that the system cannot protect the hardware when the beam intensity of the RIBF becomes higher. Therefore, we are redesigning a system by adding field-programmable gate arrays to significantly shorten the response time, rather than repeating minor improvements to save a few milliseconds.

INTRODUCTION

The RIKEN Radioactive Isotope Beam Factory (RIBF) consists of two heavy-ion linear accelerators and five heavy-ion cyclotrons. One of the linear accelerators is mainly used for experiments to search for super heavy elements, whereas the other is used as an injector to the cascades of the cyclotrons used for nuclear physics, material science, and life science applications. The cyclotron cascades can provide the world's most intense RI beams over the entire atomic mass range by using fragmentation or fission of high-energy heavy ions [1].

The components of the RIBF accelerator complex (such as the magnet power supplies, beam diagnostic devices, and vacuum systems) are controlled by the Experimental Physics and Industrial Control System (EPICS) [2] with a few exceptions, such as the control system dedicated to RIBF's radio frequency system [3]. However, all the essential operation datasets of the EPICS and other control systems were integrated into an EPICS-based control system [4]. Additionally, two types of independent interlock systems are operated in the RIBF facility: a radiation safety interlock system for human protection [5] and a beam interlock system (BIS) that protects hardware from high-power heavy-ion beams [6].

BIS OVERVIEW AND DEVELOPMENT OF NEW MACHINE PROTECTION SYSTEM

The BIS began operation in 2006, along with the beam commissioning of the RIBF. Figure 1 shows the hardware configuration and process flow of the BIS, which was developed based on Melsec-Q series programmable logic controllers (PLCs) [7]. It was designed to stop beams within 10 ms after receiving an alarm signal from the accelerator and beam line components. Upon receiving an alarm signal, the BIS outputs a signal to one of the beam choppers, which immediately deflects the beam just below the ion source. It also inserts one of the beam stoppers (Faraday cup) installed upstream of the problematic component. The BIS ignores the problems that occur downstream of the beam stopper insertion point. After inserting the relevant beam stopper, the beam chopper can be switched off, and the beam delivery can resume up to the inserted beam stopper. This feature is particularly important because if the problematic component cannot be recovered within a short time, the problem recovery time can be effectively used to readjust the beam to the inserted beam stopper. The inserted beam stopper can then be extracted from the beam line after the problem is fixed.

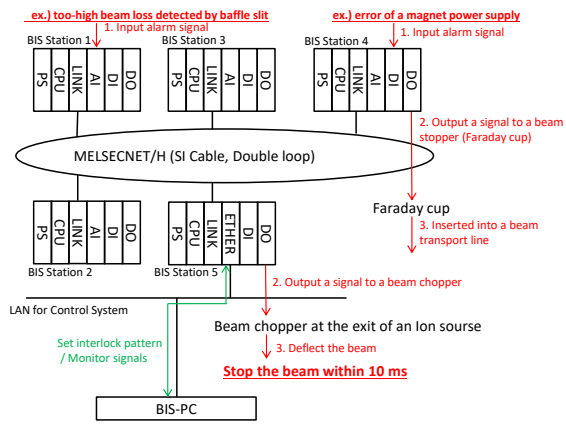


Figure 1: Example of the hardware configuration and process flow in the BIS. The green line signifies communication via Ethernet.

The BIS is still under stable operation; however, its maintenance has become gradually difficult because some

† misaki@riken.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV038

of the modules used in the system are discontinued and cannot be replaced. Additionally, the performance of the system has been declining owing to the increasing interlock signals input to the system during the 15-year developments of RIBF. The measurements performed in summer 2020 show that the average response time of the BIS, the time from when the system receives the interlock signal to when the beam is stopped is approximately 18 ms, which is greater than 10 ms, the system response time that was originally required. A response time of 10 ms or less is required to operate higher-power beams more safely in the future. Therefore, we have been developing a successor system to the BIS for a few years and attempted to apply the prototype to a small part of the facility: the AVF cyclotron and its low-energy experimental facility (AVF-BIS) in summer 2020.

The hardware constitution and process flow of the prototype and modules used in the prototype are summarized in Fig. 2 and Table 1, respectively.

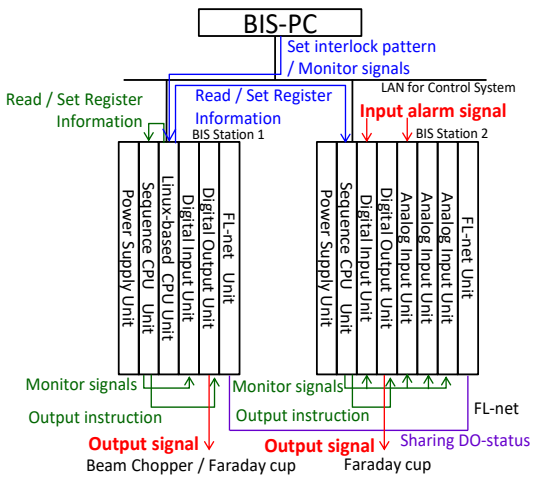


Figure 2: Hardware constitution and process flow in the prototype. Blue lines represent communication via Ethernet, green lines represent communication via PLC bus, and purple lines represent communication via FL-net.

Table 1: FA-M3 Modules Used in the Prototype

Type	Product	Note
CPU1	F3SP71-4S	Sequence CPU
CPU2	F3RP61-2L	Linux-based CPU
DI Module	F3XD32-3F	
DO Module	F3YD14-5A	Transistor contact
AI Module	F3AD08-1V	
FL-net Module	F3LX02-1N	Transmission speed: 10 Mbps

To develop the prototype, instead of adopting the Melsec-Q series PLC following the BIS, we adopted the FA-M3 series PLC, a product of Yokogawa Electric Corporation, which is widely used in RIBF control systems [8]. Details of the prototype were reported previously [9].

A new advantage of the prototype is that it utilizes two different types of central processing units (CPUs): the sequence CPU and Linux-based CPU. The former is dedicated to high-speed signal processing such as beam stop after receiving an alarm signal, whereas the latter mainly handles tasks that do not require high-speed response, such as the parameter setting of the interlock system. The introduction of the latter reduces tasks for the former, and a reduction in the response time is expected. We set up a prototype comprising two PLC stations: one with a Linux-based CPU and a sequence CPU installed in a location close to the ion sources, and the other with a sequence CPU installed in the location where wiring of various input signals was gathered. The communication between the two stations is performed through FL-net (an open network protocol used for interconnection between controllers) using dedicated wiring [10], and the signal setting and monitoring are performed by the terminal in the control room via Ethernet. In the AVF-BIS, 28 digital inputs and 23 analog inputs were used as interlock signals, and five digital outputs were used to stop the beam.

As a result of the oscilloscope-measured signal response speed in the AVF-BIS, the observed response time averaged 2 ms and 5.4 ms, respectively, within one station and with both stations. Figures 3 and 4 depict examples of the measurements, in which the yellow and green lines show the input and output signals, respectively. In Fig. 3, the signal is input to BIS station 1 in Fig. 2 and the output from BIS station 1. In Fig. 4, the signal is input to BIS station 2 and the output from BIS station 1. For both measurements, the oscilloscope was connected to BIS station 1. Therefore, in the measurement shown in Fig. 4, a cable is temporarily laid from BIS station 2 to the oscilloscope placed beside BIS station 1, and the signal is input to BIS station 2 and at the same time input to the oscilloscope via the cable.

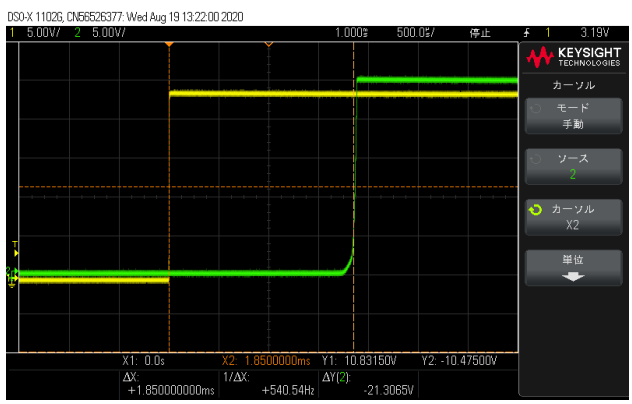


Figure 3: Signal output timing at AVF-BIS (within the same station, 1.85 ms).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV038

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

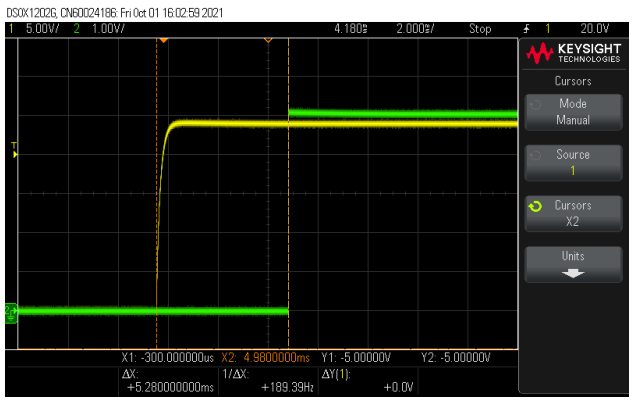


Figure 4: Signal output timing at AVF-BIS (between separate stations, 5.28 ms).

The average response time was obtained as 1.4 ms and 3.8 ms for the operation within one station and using two stations, respectively, in the test involving two stations side by side before installing the prototype as AVF-BIS [11]. The results of the two aforementioned measurements show that the signal response time increases after the prototype is installed as the AVF-BIS. There are mainly two differences before and after installing the prototype as AVF-BIS: one is that the distance between two stations increases from side by side to approximately 70 m, and the other is that the length of the program executed on the sequence CPU increases by approximately 1.2 times in the AVF-BIS as compared to the measurement in the prototype. However, these reasons are not sufficient to explain the difference in the response time, and details of the cause are currently under investigation.

The measurement result of the response time was sufficient for the performance of the AVF-BIS. However, scaling up the prototype to a system consisting of five PLC stations with approximately 400 signals, similar to the existing BIS, it is estimated that the response time of the system will be long because the transmission time varies depending on the number of stations. According to the data released by Yokogawa Electric Corporation, when the number of stations in the system increases from two to five, the signal transmission time is almost doubled under the condition that the size of the transmitted data is the scale of the AVF-BIS. Because the ratio of the transmission time to the response time between two stations is under investigation and is not currently clear, it is not possible to accurately estimate the response time of the system when the AVF-BIS is expanded to five stations. However, we reckoned that it was necessary to consider the possibility that the performance required for the system could not be achieved with a sufficient margin in the expansion of the current system using the sequence CPU and FL-net. Therefore, we began to redesign the system.

To reduce the signal transmission time in the stations, we opted to use a twisted cable. Additionally, to further shorten the response time, we plan to introduce a field-programmable gate array (FPGA) to process interlock signals as fast as possible in the new system. Because the signal propagation speeds of the twisted cables range from 4.5 to 5.5 ns/m, the response speed of the new system is expected to be an order of magnitude faster than the existing BIS.

DEVELOPMENT OF FPGA-BASED MACHINE PROTECTION SYSTEM

In the development of a BIS system using FPGAs (hereinafter, FPGA-BIS), the method of stopping a beam to protect the machine is basically the same as that of the BIS and AVF-BIS. In the AVF-BIS, which has been improved based on the operational experiences of the BIS, the system response time was successfully reduced using two types of CPUs according to the required processing speed. Therefore, we also properly use FPGAs and CPUs according to the signal processing time required in FPGA-BIS, which will be used to evaluate various interlock conditions imposed on each input signal as fast as possible. For example, we require a certain duration time for some interlock signals to avoid false alarm signals originating from the malfunction and/or noises of the system. Furthermore, the BIS should stop the beam only when an alarm signal is output from the device upstream of the beam stopper inserted in the beam line. By executing these condition judgements on the FPGA, we expect that the high-speed processing of signals in the system is realized. However, CPUs constantly execute processing such as setting signal parameters and monitoring signals indicating the state of the beam stopper. As in the case of the AVF-BIS, the EPICS will be executed on the CPU.

Each station that composes the FPGA-BIS will be installed at or near the existing BIS station to maximize the reuse of the existing signal wiring of the BIS. Therefore, the stations are widely distributed in the facility as well as in the BIS.

To achieve the performance required in FPGA-BIS, we are now studying two systems as candidates: FA-M3 series PLC by adding input/output (I/O) modules with an FPGA (FPIO module) [12], and CompactRIO, a product by National Instruments [13]. The FPIO module (F3DF01-0N) was equipped with an FPGA and had 24 input contacts and 24 output contacts. The advantages and disadvantages of each system are as follows.

- The advantage of the FA-M3 PLC system is that it is familiar with its operation because it is widely used in the RIBF control system. Additionally, because each FPIO module has an FPGA, the processing of the input signal can be closed in the module, thus, the system can be expanded simply by adding the FPIO module (there is no need to coordinate with other FPIO modules).
- The disadvantage of the FA-M3 PLC system is that the total cost increases. The FPIO module is more expensive than a normal I/O module; however, many FPIO modules are required because all alarm and output signals that require high-speed processing need to be connected to the FPIO module. Another concern is the duration of maintaining the present FPIO module that uses Spartan 6 FPGA, the release of which was more than 10 years ago.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV038

- The advantage of the CompactRIO system is that the total cost can be reduced (estimated to be approximately half that of the FPIO system). In the CompactRIO system, an FPGA is mounted on the chassis instead of individual modules, thus, normal I/O modules can be used to connect the signals even if they require high-speed processing. Additionally, LabVIEW can be used to develop FPGA logic [14], and even if the FPGA is updated to a new one, it is not necessary to recreate the logic once it is developed on LabVIEW because it is expected to absorb the difference between different FPGA versions. LabVIEW is widely used in various systems at the RIBF, so it is familiar to us as well as the FA-M3 PLC system. Furthermore, the machine protection system using CompactRIO has been in operation for more than 10 years at one of the core experimental facilities of the RIBF and can be used as a reference, although the system scale is different [15].
- The disadvantage of the CompactRIO system is that it is necessary to recompile the FPGA logic whenever the number of signals handled by the system increases and new modules are added.

We are currently preparing to build prototypes for both the systems. After testing both prototypes, we selected the FPGA-BIS. We aim to replace the BIS with the FPGA-BIS within 2 years.

REFERENCES

[1] O. Kamigaito *et al.*, "Recent progress in RIKEN RI Beam Factory", in *Proc. Cyclotrons2019*, Cape Town, South Africa, Sep. 2019, paper MOB01, pp. 12–16. doi:10.18429/JACoW-Cyclotrons2019-MOB01

[2] EPICS, https://epics.aps.anl.gov

[3] M. Komiyama *et al.*, "Recent improvements to the RIKEN RI Beam Factory control system", in *Proc. PCaPAC2016*, Campinas, Brazil, Oct. 2016, paper WEPOPRP011, pp. 31–34. doi:10.18429/JACoW-PCaPAC2016-WEPOPRP011

[4] A. Uchiyama, M. Komiyama, M. Kidera, and N. Fukunishi, "Data archive system for super conducting RIKEN linear accelerator at RIBF", in *Proc. IPAC2021*, Campinas, SP, Brazil, May. 2021, paper TUPAB297, pp. 2178–2181. doi:10.18429/JACoW-IPAC2021-TUPAB297

[5] H. Sakamoto *et al.*, "HIS: The radiation safety interlock system for the RIKEN RI Beam Factory," RIKEN Accel. Prog. Rep. vol. 37, pp. 281, 2004.

[6] M. Komiyama, M. Fujimaki, I. Yokoyama, and M. Kase, "Status of control and beam interlock system for RARF and RIBF," RIKEN Accel. Prog. Rep. vol. 39, pp. 239, 2006

[7] Melsec-Q, https://www.mitsubishielectric.co.jp/fa/products/cnt/plcq/items

[8] FA-M3, http://www.FA-M3.com/jp

[9] M. Komiyama *et al.*, "Recent updates of the RIKEN RI Beam Factory control system", in *Proc. ICALEPCS2019*, New York, NY, USA, Oct. 2019, paper MOPHA073, pp. 384–387. doi:10.18429/JACoW-ICALEPCS2019-MOPHA073

[10] FL-net, https://www.jemanet.or.jp/Japanese/standard/open

[11] M. Komiyama *et al.*, "Recent update to the RIKEN RI Beam Factory control system", in *Proc. ICALEPCS2017*, Barcelona, Spain, Oct. 2017, paper TUPHA028, pp. 427–430. doi:10.18429/JACoW-ICALEPCS2017-TUPHA028

[12] Yokogawa Electric corporation, "Catalog of F3DF01-0N", unpublished.

[13] CompactRIO, https://www.ni.com/ja-jp/shop/compactrio.html

[14] LabVIEW, https://www.ni.com/ja-jp/shop/labview.html

[15] K. Yoshida, "Fast beam interlock system for BigRIPS separator," RIKEN Accel. Prog. Rep. vol. 52, pp. 131, 2019.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV039

NOVEL PERSONNEL SAFETY SYSTEM FOR HLS-II*

Z. Huang, K. Xuan, C. Li, J. Wang, X. Sun, S. Xu, G. Liu†, NSRL, USTC, Hefei, Anhui 230029, China

Abstract

The Hefei Light Source-II (HLS-II) is a vacuum ultraviolet synchrotron light source. The Personnel Safety System (PSS) is the crucial part to protect staff and users from radiation damages. In order to share access control information and improve the reliability for HLS-II, the novel PSS is designed based on Siemens redundant PLC under EPICS environment which is composed by the safety interlock system, access control system and the radiation monitoring system. This paper will demonstrate the architecture and the specific design of this novel PSS and shows the operation performance after it has been implemented for 2 years.

INTRODUCTION

The Hefei Light Source-II (HLS-II) is a dedicated synchrotron radiation facility which can provide radiation from infrared to vacuum ultraviolet with both Top-Off and Decay operation modes [1].The Personnel Safety System (PSS) is the crucial part of the HLS-II to keep radiation damage away from the staffs and users. The previous PSS in HLS-II was developed by Simens SiPass system which lacked of personnel management function and it's hard to share information. For solving these disadvantages, the novel PSS has been designed for HLS-II.

The Programmable Logic Controller (PLC) and redundant technology are wildly used in the PSS design of big scientific facilities to meet the requirements of the high reliability and stability, such as the European Spallation Source (ESS) [2] and the High Intensity D-T fusion neutron generator (HINEG) [3]. The novel HLS-II PSS is designed based on the Siemens redundant PLC S7-412-5H under the Experimental Physics and Industrial Control System (EPICS). EPICS is a set of open-source software tools, libraries, and applications that are widely used in big scientific facilities [4, 5]. The novel HLS-II PSS contains 3 parts: the safety interlock system, the access control system and the radiation monitoring system. The safety interlock system is used to define the interlock logic to be implemented, the access control system is designed to restrict the access of the staffs and users at HLS-II and provide the personnel management function, and the radiation monitoring system is used to monitor the dose rate in the light source and the surrounding areas.

In this paper, section II introduces the system architecture of the novel HLS-II PSS, section III gives the details about the design of the safety interlock system and the personnel management function in the access control system, and section V shows the operator interfaces (OPIs) design and the operation situation of the novel HLS-II PSS.

* Work supported by National Natural Science Foundation of China (No.11375186)
† Corresponding author, gfliu@ustc.edu.cn

SYSTEM ARCHITECTURE

The novel HLS-II PSS ensures the personal safety by monitoring the radiation dose rate, controlling interlock signals and executing interlock actions. We integrate the safety interlock system, the access control system and the radiation monitoring system into EPICS environments for information sharing. Meanwhile, we can use the existing data archiver and alarm toolkits provided by the EPICS community to archive the historical data and publish the alarm information [6]. The system architecture of the novel HLS-II PSS consists of 3 layers: the EPICS layer, the controller layer and the devices layer as Fig. 1 shown.

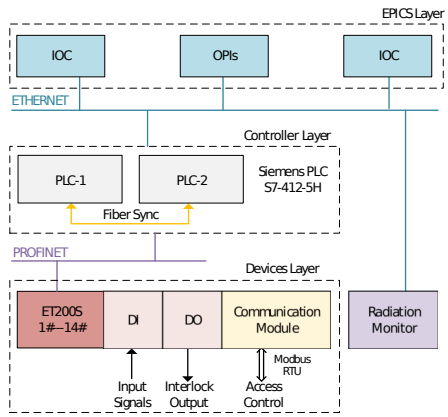


Figure 1: Architecture of the Novel HLS-II PSS.

In the controller layer, there is only one pair of Siemens redundant PLC S7-412-5H. PLC can gather IO signals and access data from 14 IO stations by fiber optic cables, and receive the radiation monitoring signal and commands from other EPICS IOC. The redundant PLC pair has two high performance PLCs that are backed up with each other, one of them takes the role of MASTER PLC and the other one takes the role of SLAVE PLC. During the operation, those two PLCs synchronise programs and real-time data over a high-speed fiber, and the roles between two PLCs can switch over when the MASTER PLC fails.

In the devices layer, 14 Siemens ET200S IO stations are distributed nearby the 14 security doors. The input signals of search buttons, emergency buttons and security doors are collected into the IO stations through the DI modules. The signals of the audible and visual alarm devices, button lamps and the interlock actions are outputted through the DO modules. In addition, the data of card reader is transmitted into the IO station via Modbus-RTU protocol.

According to the design, there are 134 input signals monitored, including 41 search buttons signals, 25 emergency stop buttons signals, 42 security doors signals, and 26 radiation monitoring signals. All the signals are from IO stations, except the radiation monitoring signals. The communication

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV039

between PLC and IO stations adopts PROFIENT Real-Time protocol, the communication cycle time is 2 ms. The communication between IOC and PLC uses Ethernet with a communication cycle time of 100 ms. Radiation monitoring signal is used to monitor the average of the radiation dose in 5 seconds, so the real-time performance for this signal is not high. It is transmitted between IOCs over EPICS Channel Access protocol.

DESIGN OF THE SAFETY INTERLOCK SYSTEM AND ACCESS CONTROL SYSTEM

The radiation monitoring system has been developed based on EPICS and deployed at HLS-II in 2017 [7]. Therefore this section focuses on the design details of the safety interlock system and the access control system.

Safety Interlock System

The safety interlock system is the crucial system to process the interlock logic. Figure 2 shows the workflow chart of the safety interlock system and Table 1 shows the definition and actions of all states. During the operation of the safety interlock system, there are 3 operation states: the released state, the searching state and the interlocked state.

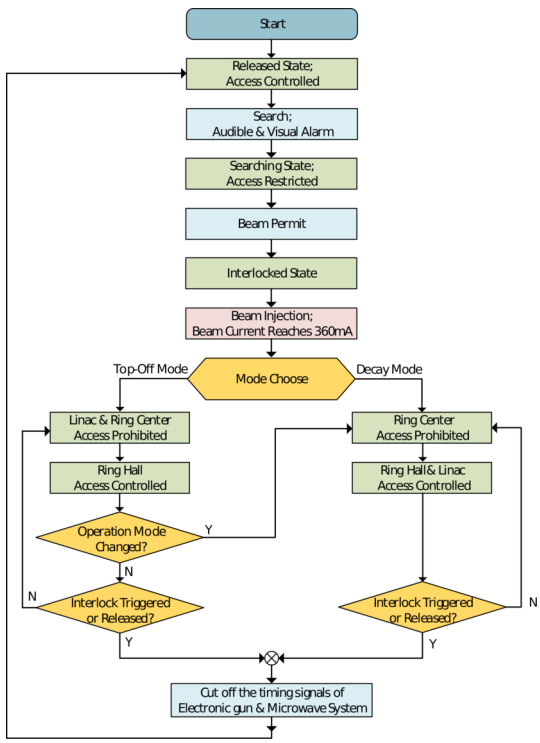


Figure 2: Workflow of the Safety Interlock System.

When the facility is turned on, the safety interlock system enters into the released state. At HLS-II, there are 3 safety interlock areas: the linac, the ring center and the ring hall. In the released state, the interlock system of the 3 areas should be released. All the interlock signals are reset and the search lamps are turned off.

Table 1: States of the Safety Interlock System

States	Definition	Actions
Released State	Release the interlock	Reset interlock logic Turn off search lamps
Searching State	To establish the interlock	Audible and visual alarm Search the interlock areas Turn on search lamps Lock the security doors
Interlocked State	Interlock is established	Beam Injection Operation mode choose

The searching state is aimed to establish the interlock. When the searching state starts, the operator should inform all person to leave the interlock areas by audible and visual alarm, then the operator searches the interlock areas in the order of the linac, ring center and ring hall. If all the search buttons are pressed and the security doors is locked, the operator can give the beam permit command to transit the operation state into the interlocked state.

For the HLS-II, after the beam is injected and the beam current of storage ring reaches 360 mA, there are 2 types of operation modes to choose: Top-Off mode and Decay mode. Top-Off mode is a high performance operation mode, and the beam is injected in every few minutes [8]. Thus the linac and the ring center should be kept in the interlocked state in Top-Off mode. For Decay mode, the beam is injected in every few hours, the interlock of the linac and the ring hall can be released while the ring center must be kept in the interlocked state.

During the interlocked state, if the interlock system is released by the operator or triggered by the interlock signals, such as the emergency buttons is pressed or the radiation monitoring system detects excessive radiation dose, the safety interlock system will cut off the timing signals of the electronic gun and the microwave system, and transit into the released state.

Access Control System

The design of the access control system at HLS-II is based on the principle of classified protection. The interlock areas can be classified into 3 kinds according to the radiation dose rates: the high radiation area, the radiation area and the safety area. The radiation dose rate of the high radiation area is more than 10 μSv/h, of the radiation area is between 1 μSv/h and 10 μSv/h, and of the safety area is less than 1 μSv/h. Table 2 shows the safety classification of the interlock areas at the different PSS state. For the high radiation area, the access of security doors is prohibited, and no personnel is allowed to open the doors under any conditions; for the radiation area, only the staffs have the restricted access authority; and for the safety area, staffs, users and all other personnel can open the security doors with the access card.

At HLS-II, the security doors must be opened by access card. When access card is swiped, the card information

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV039

Table 2: Safety Classification of the Interlock Areas at HLS-II

Operation States		Linac	Ring Center	Ring Hall
Released State		Radiation Area	Radiation Area	Safety Area
Searching State		Radiation Area	Radiation Area	Radiation Area
Interlocked State	Beam Enable	High Radiation Area	High Radiation Area	High Radiation Area
	Top-Off Mode	High Radiation Area	High Radiation Area	Safety Area
	Decay Mode	High Radiation Area	Radiation Area	Safety Area

will be sent to the PLC. Then the PLC judges the access permission according to the access authorities for each card and the safety classification of the interlock areas. The card number and authentication information are sent to IOC and archived in EPICS database as an event for recording and tracking. Meanwhile, the OPIs can display the access state synchronously.

In the access control system, the personnel count function is realized in the EPICS IOC by state notation language (SNL). SNL is a domain specific programming language which can provide a simple yet powerful tool for sequential operations in a real-time control system. According to the card number and authentication information, the SNL program counts the personnel number for each interlock areas, and writes the count results into the EPICS records.

OPI DESIGN

The OPIs of the novel HLS-II PSS is composed of the OPI of the linac area and the OPI of the storage ring area, as shown in Figs. 3 and 4. The linac area includes the linac tunnel and transport line tunnel. The storage ring area includes the ring center and the ring hall.

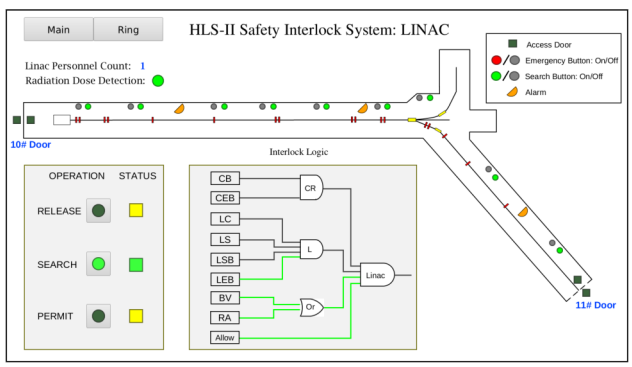


Figure 3: The OPI of Linac.

The OPI can be divided into four parts: 1) the distribution and state of the security doors, the emergency buttons, the search buttons and the alarm devices; 2) the personnel count and the radiation dose monitoring state for the interlock areas; 3) the interlock logic diagram; 4) the operation commands and the related states. The line color in the interlock logic diagram is variable depending on the logic value, it is gray when the logic value is 0, and green when the logic value is 1.

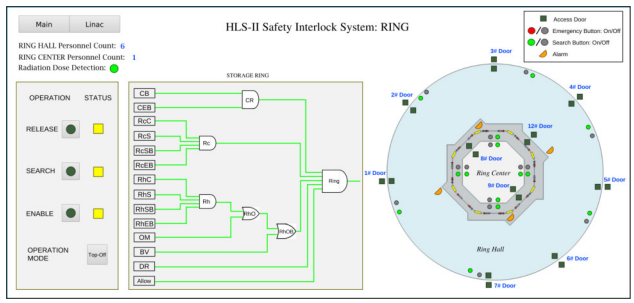


Figure 4: The OPI of Storage Ring.

OPERATION STATUS

The development of the novel HLS-II PSS is completed. To ensure that this system could work as designed, we set up the offline platform and continuously tested for one month. The test results demonstrate that all the devices can run well and stably, the interlock actions can be executed effectively, and the Top-Off mode and Decay mode can be supported successfully. Meanwhile, the offline platform can work properly during the redundant PLCs switch over.

The development of the novel HLS-II PSS is completed and it's deployed during 2019 summer. Figure 5 is the photo of the Door Display OPI which is located beside the No.7 security door. Distinct from the OPIs in Linac area and storage ring area, this OPI will shown the operation modes and access state as well. When the access card is swiped, it can alarm the access result via audio and text at the meantime.

During the 2 years operation, the novel HLS-II PSS keeps running stably behind the control of redundant PLCs. The safety interlock system can executed interlock actions precisely and all 3 interlock states can be switched timely . It also performs well on supporting both of Top-off mode and Decay mode. The new access control system improves the effectivity of personnal management, and the radiation system can be integrated into this novel PSS smoothly. All the data in novel PSS can be shared and processed real-timely under EPICS environment.

CONCLUSION

In the novel design of HLS-II PSS, the safety interlock system, the access control system and the radiation monitoring system are integrated into EPICS environments, it is easy to share information and use the existing toolkits provided by the EPICS community. In the access control

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV039

Figure 5: Photo of the novel HLS-II PSS Door Display OPI.

system, the personnel management function is designed for monitoring the entry and exit of staffs and users to meet the HLS-II management requirements. After 2 years successful operation from 2019, the novel HLS-II PSS has proofed its highly reliability and stability which can fulfull the design requirements.

REFERENCES

[1] L. Wang, W. Li, G. Yao, *et al.*, "The upgrade project of Hefei light source (HLS)", in *The 1st International Particle Accelerator Conference, IPAC'10*, Kyoto, Japan, 2010, pp. 2588–2590.

[2] M. Mansouri, S. Birch, A. Nordt, *et al.*, "Accelerator Personnel Safety Systems for European Spallation Source", in *Proc. The 8th International Particle Accelerator Conference, IPAC'17*, Copenhagen, Denmark, 2017, pp. 1884–1886.

[3] W. Wang, Y. Song, J. Wang, *et al.*, "Design of the personnel radiation safety interlock system for high intensity D–T fusion neutron generator", *Journal of Fusion Energy*, vol. 34, pp. 346-351, 2015.

[4] L. Han, Y. Chen, J. Cai, *et al.*, "The application of EPICS in TMSR radiation protection and access control system", *Nuclear Science and Techniques*, vol. 27, p. 41, 2016.

[5] D. Sanz, M. Ruiz, R. Castro, *et al.*, "Implementation of Intelligent Data Acquisition Systems for Fusion Experiments Using EPICS and FlexRIO Technology", *IEEE Transactions on Nuclear Science*, vol. 60, no. 5, pp. 3446–3453, 2013.

[6] Z. Hu, Q. Mi. L. Zhen, Z. Li, *et al.*, "EPICS data archiver at SSRF beamlines", *Nuclear Science and Technology*, vol. 25, p. 020103, 2014.

[7] Z. Zeng, J. Wang, K. Xuan, *et al.*, "Dose Rate Monitor System for the HLS-II. Dose Rate Monitor System for the HLS-II", *Nuclear Electronics and Detection Technology*, vol. 37, pp. 43–46, 2017.

[8] W. Xu, D. Jia, S. Jiang, *et al.*, "Upgrade Project on Top-Off Operation for Hefei Light Source", in *The 8th International Particle Accelerator Conference, IPAC'17*, Copenhagen, Denmark, 2017, pp. 2719–2722.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV040

DESIGN OF MACHINE PROTECTION SYSTEM FOR SXFEL-UF

Chunlei YU, Jianguo DING, Huan Zhao
Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, P.R China

Abstract

Shanghai Soft X-ray Free-Electron Laser (SXFEL) facility is divided into two phases: the SXFEL test facility (SXFEL-TF) and the SXFEL user facility (SXFEL-UF). SXFEL-TF has met all the design specifications and has been available in beam operating state. SXFEL-UF is currently under commissioning and is planned to generate 3 nm FEL radiation using a 1.5 GeV electron LINAC. To protect the critical equipment rapidly and effectively from unexpected damage, a reliable safety interlocking system needs to be designed. Machine Protection System (MPS) is designed by Programmable Logic Controller (PLC) and Experimental Physics and Industrial Control System (EPICS) which is based on a master-slave architecture. In order to meet different commissioning and operation requirements, the management and switching functions of eight operation modes are introduced in the MPS system. There are two FEL lines in user facility named SXFEL beamline project (BSP) and undulator (UD) , and the corresponding design of MPS is completed. This paper focuses on the progress and challenges associated with the SXFEL-UF MPS.

INTRODUCTRION

Shanghai Soft X-ray Free-Electron Laser test facility(SXFEL-TF) has been successfully completed in 2020, and the beam energy of the test facility is 840 MeV. The SXFEL user facility (SXFEL-TF) is a critical development step toward the construction of a soft X-ray FEL user facility in China and has been currently undergoing commissioning at the Shanghai Synchrotron Radiation Facility (SSRF) campus[1]. The LINAC accelerator of SXFEL-UF is designed to increase the beam energy to 1.5GeV[2]. Not only the original undulator beam line has been upgrade in SXFEL-UF, but also a new undulator beam line adopts high-throughput working modes such as SASE has been new built.

The requested response time of Machine Protection System (MPS) is less than 20ms, so PLC is employed to execute the underlying logic. Experimental Physics and Industrial Control System (EPICS) is a set of software tools for building distributed control system to operate devices such as particle accelerators and large experiments[3]. The EPICS framework is adopted in the control system in SXFEL-UF and MPS. Due to the hardware structure and function division for test facility, modification and extension has been implemented for new demands of SXFEL-UF.

MPS COMPOMENT

System Structure

The structure of SXFEL machine protection system is shown in Figure 1. The operator interface (OPI) is connected to the input and output controller (IOC) through the local area network, and takes a approach of the EPICS CA protocol. IOC runs on Linux system, and MOXA DA-662 embedded computer is used as IOC server, equipped with 16 serial ports and 4 network ports. Embedded software and hardware technology is applied to complete the development of embedded EPICS IOC.

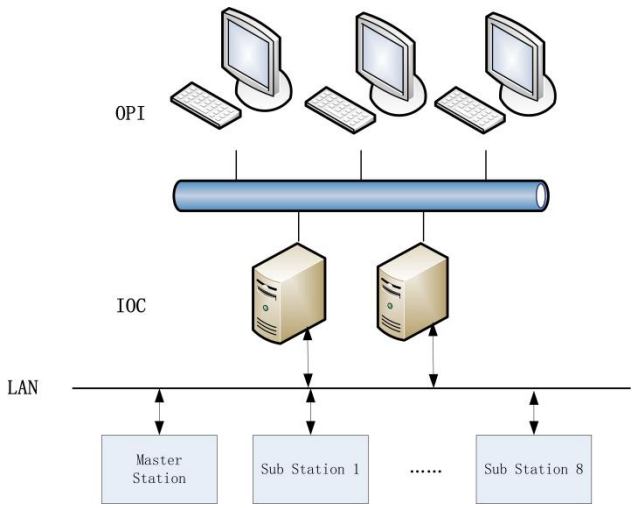


Figure 1: The scheme of SXFEL-UF MPS.

Signal Statistics

Almost all the systems should be assigned the interfaces with MPS, such as radio frequency system, vacuum system, power supply system, water cooling system, timing system and personal safety protection system (PPS) and so on. At present, the number of interlocking input/output signals is aboout 1000, the main equipment interlocking signals are listed in Table 1.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV040

Table 1: Summary of Main Interlock Signals

No.	device/signal	number
1	timimg control	17
2	vacuum warn and alarm	234
3	pump power supply	302
4	vacuum valve status and control	135
5	modulator status and control	51
6	PPS signal	12
7	cooling water status	78
8	RF switch signal	8
9	Leakage/pressure monitor signal	24

Function Realization

Omron CS1G-H series PLC are applied in MPS, and the CPU model is CPU43. There were 7 PLC stations used for master-slave architecture in SXFEL-TF, including one station for injector (IN), 3 stations for Linac (LA), 2 stations for undulator (UD) section, and one master station for comprehensive treatment. In SXFEL-UF MPS, 9 stations have been arranged for total interlock processing together, which the master station, injector station and 3 linac stations are reserved, but more signals and logic processing are increased. Transmission line (TL) are new added before undulator lines in user facility, so one station for TL and 3 UD stations used for both undulator lines have been restudied and implemented in our new design.

Master station is provided with the function to management and switching the operation mode, and more important function is can synthesize the selected mode and interlock input signals to make corresponding logical judgments and responses. One of the features is to logically integrate the global interlock signal and the output signal from each slave station and fan out to other slave stations. Another feature is to permit the global output signals, the exciting permit signals such as:

- Beam permit.
- Timing system permit signals.
- Driving laser shutter.
- PPS and shutter permit.
- Undulator line selection.
- Operation mode switching.

The local sub stations are connected with device signals such as vacuum, valves, water cooling, air pressure, and water leakage of the accelerator section where they are located. Every sub-station operates at different interlocking logic in accordance with different interlocking requirements to ensure the normal operation of the equipment in its own section. And sub-stations also send the beam permit signal and timing control output signal to master station through synthesis all the logic signals.

In addition, SXFEL has been arranged by a total of 17 modulators. According to the interlock signals of the section where the modulator is located and the command from the Master station, some of the sub-stations need to complete the judgment of the modulator interlock status and control the operation of the modulator

Another significant function of the local sub-station is to realize the local vacuum protection. When there is a vacuum leakage, the vacuum isolation is realized through valve control to avoid leakage and spreading. The vacuum alarm signal needs to be transmitted between the sub-stations, and the upstream station transmits the end vacuum status signal to the downstream sub-station as the vacuum protection input signal. The main signal interaction between each stations shown in figure 2.

OPERATION MODE DESIGN

In the early stage of beam debugging, RF conditioning and section commissioning of accelerator are required. In order to facilitate the operation and reduce the occurrence of misoperation, mode control and switching have been realized in MPS according to different accelerator operating conditions. The adjustment personnel can select a specific mode according to the specific conditions of debugging, and MPS can automatically match different interlock logic in different modes.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

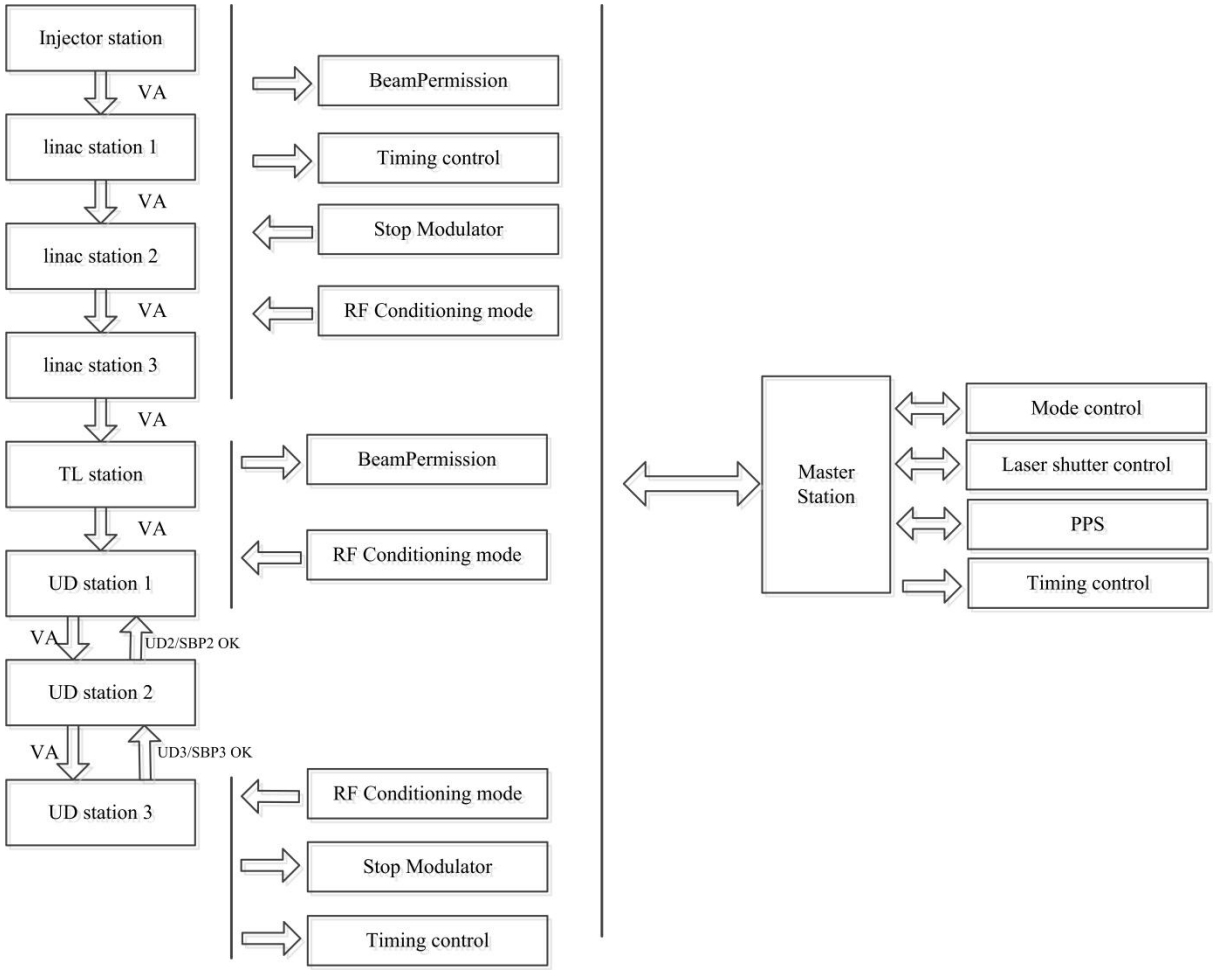


Figure 2: The structure of stations and signal interaction.

7 operation modes are realized in SXFEL-UF MPS, which continuing the original design in test facility. Two beamlines can be chosen flexibely, and the beamline selection is achieved individually.

- STOP mode. The driving laser shutter should be shut down, all timing trigger signals should be stop and all vacuum valves are closed.
- RF Conditioning mode. it is necessary to stop the timing control for the driving laser and close the laser shutter, and the timing trigger to all power sources should be maintain in the normal interlock state.
- Injector debug mode. In this mode, only debugging of the injector section is performed. So the interlock for injector section is normal, but signals from other sections such as linac or undulators are bypassed.
- Linac debug mode. It is necessary to establish a normal interlock from the injector section to the end of the main accelerator, and bypass the interlock input signals from the undulator section.
- Undulator debug mode. Establish normal interlock from the injector section to the end of the main accelerator, but the signal of the B iron at the end of UD should be bypassed.
- Running mode. MPS should operate normally in accordance with the predetermined interlock logic.
- Laser debug mode. All timing triggers should be stop except the trigger for laser system.

The operation mode control is realized by hardware, and the control of the mode panel and the specific logic switching are all achieved by the master station. The design of the mode control panel is shown in Figure 3. The operator needs the key to activate the mode selection function, and the key can be switched between "mode lock" and "mode selection". After entering the "mode selection", 7 modes can be cyclic selected by clicking the "selection button". After the selection is completed, the key should be turned to "mode lock" to complete the switching. The master station also sends effective information to each sub-station and device after the mode is locked, and then the entire system can perform the interlock logic under the new mode.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV040

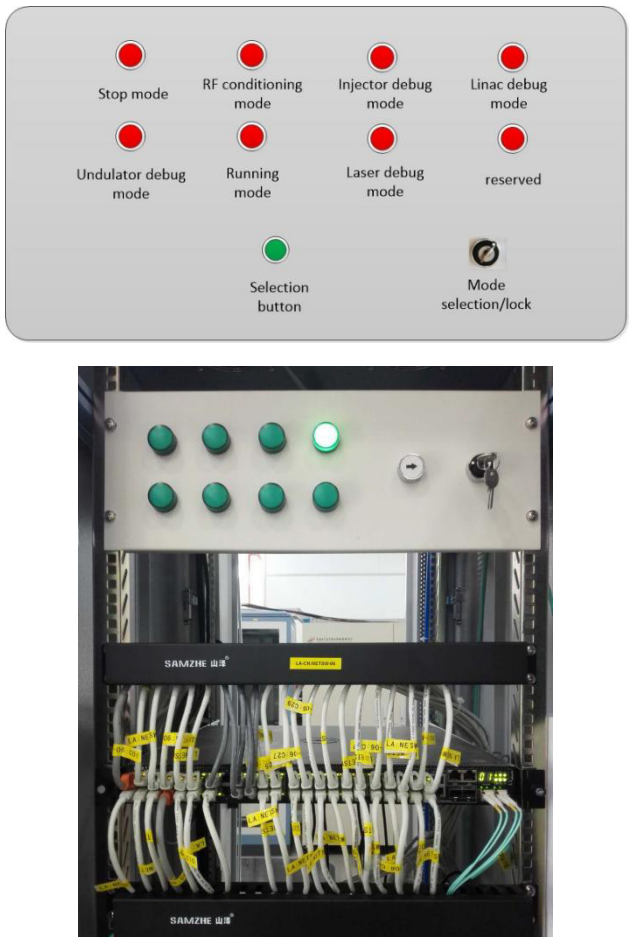


Figure 3: The design of control panel for mode selection.

ALERTING SIGNAL PROCESSING

Once the equipment involved in the interlock fails or the signal exceeds the set threshold, MPS will take protective measures according to the predetermined interlock logic and output the corresponding interlock protection signal. When the alerting signal returns to normal, the interlock output signal is restored or not at this time, which depends on whether the interlock signal is latched. Once the fault signal is latched the alarm signal must be confirmed by the operator before the interlocking state can be canceled, even if the input signal returns to normal, and it cannot be automatically reset.

The alerting signals should be latched and recorded when the failures occur at the normal running mode or debugging mode in SXFEL-UF MPS. But the state of each system can not meet the operating standard at RF conditioning mode, the latch function may cause frequent operation for reset by the operator. So the automatic reset method is adopted to eliminate the interlock output. The Master station will send a signal to all the sub-stations whether to work at the RF conditioning mode. According to this, each sub-station will determine whether the local signals needs to be latched. For each PLC station, every interlocking input signal corresponds to a latch signal and a reset signal. The processing flow of the SXFEL-UF MPS input signals is shown in Figure 4.

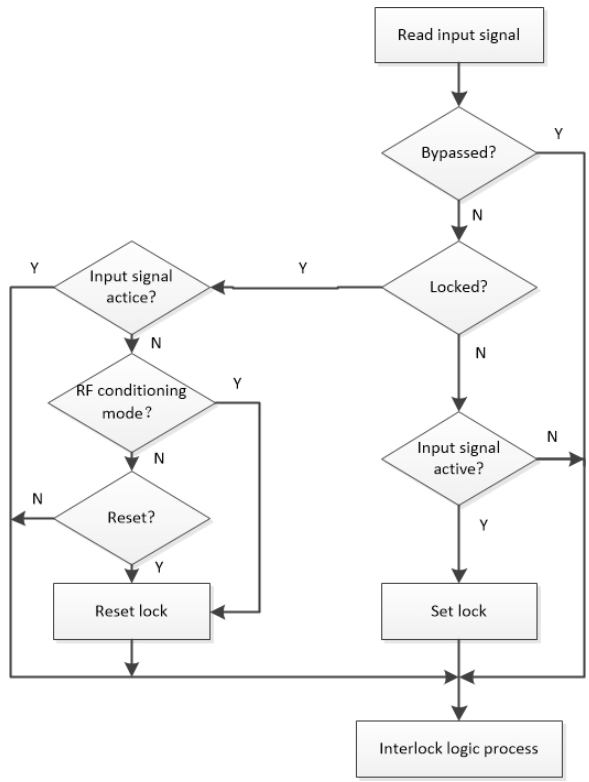


Figure 4: Flow chart of input signal process.

OPERATION GUI

SXFEL control system base on EPICS, and MPS remote control system uses MOXA DA-662 embedded sever built-in Monta Vista Linux system as EPICS IOC. The netDev IOC device driver developed by Japan KEK is used, which can realize the communication with OMRON CS1 series PLC through Ethernet module[4].

The operator interface (OPI) is developed by EDM, which is composed of the main interface of MPS and various sub-interfaces. 6 new sub interfaces for both undulator lines have been added, and the two lines selection and enable judgment have been finished and modified at main interface for SXFEL-UF. The global interface of SXFEL-UF MPS shown in Figure 5, and the interface of MPS for SXFEL injector is shown in Figure 6.

The current operating mode of the accelerator, the interlock status of each station, and overview information of MPS signals such as vacuum and water cooling are all displayed in main interface. At the interface of each accelerator section and undulator section, the specific information of the interlock input signals can be obtained at the sub interface divided by accelerator section. Also the corresponding control operations for reset, bypass and equipment control can also be performed.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV040

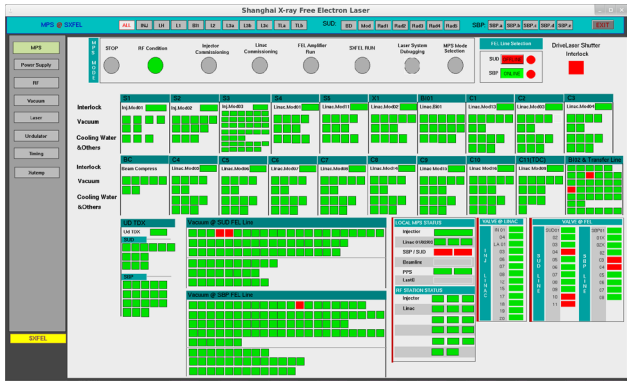


Figure 5: The global interface of SXFEL-UF MPS.

Figure 6: The interface of interlock system for SXFEL injector.

CONCLUSION

PLC is used to achieve the collection, calculation and interlocking protection of all interlocking signals at SXFEL-UF MPS. Based on the architecture of Master main station and local sub-station design, interlocking protection with operation mode switching and management functions is realized. The interlocking protection requirements of various systems such as vacuum, water cooling, high frequency, timing and personal safety have been available in our system. And distributed control system based on the standard model and EPICS have been build and realized for SXFEL-UF MPS system.

Through online testing, the response time of the system is less than 15 ms, which totally meets the physical requirements of 100 ms response time at the device repetition frequency of 10 Hz. The performance may also satisfy the demand of 20 ms response time .

After a long-term operation measurement, the SXFEL-UF MPS is stable and reliable, and the interlocking function is running normally, which effectively meets the requirements of debugging and operation at various stages.

REFERENCES

[1] C.C Xiao, J.Q Zhang, et al. “Design and preliminary test of the LLRF in C band high-gradient test facility for SXFEL”, Nucl. Sci. Tech., vol. 31, no. 100, p. 99, Oct. 2020.

DOI: 10.1007/s41365-020-00806-6

[2] C.L Yu, H Zhao, J.G D. “Interlock System for Soft X-ray Free-electron Laser in Shanghai ”, Atomic Energy Science and Technology, vol. 52, no. 4, p. 756, Apr. 2018.

[3] EPICS, http://www.aps.anl.gov/epics

[4] netDev, http://www.linac.kek.jp/cont/epics/netdev

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV041

IMPLEMENTATION OF A VHDL APPLICATION FOR INTERFACING ANYBUS CompactCom

S.Gabourin*, S. Pavinato, A. Nordt, European Spallation Source, Lund, Sweden

Abstract

The European Spallation Source (ESS ERIC), based in Lund (Sweden), will be in a few years the most powerful neutron source in Europe with an average beam power of 5 MW. It will accelerate proton beam pulses to a Tungsten wheel to generate neutrons by the spallation effect. For such beam, the Machine Protection System (MPS) at ESS must be fast and reliable, and for this reason a Fast Beam Interlock System (FBIS) based on FPGAs is required. Some protection functions monitoring slow values (like temperature, mechanical movements, magnetic fields) need however less strict reaction times and are managed by PLCs.

The communications protocol established between PLCs and FBIS is PROFINET fieldbus based. The Anybus CompactCom allows an host to have connectivity to industrial networks as PROFINET. In this context, FBIS represents the host and the application code to interface the AnyBus CompactCom has been fully developed in VHDL.

This paper describes an open source implementation to interface a CompactCom M40 with an FPGA.

INTRODUCTION

The European Spallation Source (ESS), an accelerator driven research facility located outside of Lund, Sweden, is currently in its construction and early operation phase, and aims to be the most powerful and bright neutron source in the world by 2025. ESS is a long-pulse neutron source, and consists of a 600 m long proton LINAC, a rotating helium-cooled tungsten target, creating neutrons through the spallation process and 22 different neutron beam ports, equipped with neutron scattering research instruments. The unique time structure of long neutron pulses (2.86 ms) at low frequency (14 Hz) will significantly expand the possibilities for neutron science to probe material structures and dynamics [1]. The proton beam power of 125 MW per pulse (5 MW average) will be unprecedented and its uncontrolled release can cause serious damage of equipment within a few microseconds only. To maximize operational efficiency of ESS, allowing for very high beam availability with high reliability towards the end-users, accidents shall be avoided and interruptions of beam operation have to be minimized and be limited to a short time. Finding an optimum balance between appropriate protection of equipment from damage and high beam availability is the key principle on which the ESS Machine Protection Strategy is being based on [2]. Implementing and realizing the measures needed to provide the correct level of protection in case of a complex facility like ESS, requires a systematic approach, enabling seamless integration of the several 100 protection functions that span over multiple systems. The entity performing the final logic on whether beam operation is allowed or needs to be interrupted, is called Beam Interlock System (BIS), and consists of four PLC based interlock systems and the FPGA based Fast Beam Interlock System (FBIS). The FBIS takes the ultimate decision on safe beam operation and is the only system that can trigger a beam stop. It is designed to stop beam production within 3 µs for the fastest failures at a safety integrity level of SIL2 according to the IEC61508 standard. These requirements result from a hazard and risk analysis being performed for all systems at ESS. The complexity of the ESS machine (multiple beam destinations, beam modes, etc) requires not only transferring binary data from the PLC based interlock systems to the FPGA based Fast Beam Interlock System (Beam Permit OK/NOK), but also to transfer information on e.g. machine configuration, device location, etc.. For that purpose, a so-called datalink has been implemented. It transmits data from the PLCs via PROFINET towards the FPGAs, using an intermediate commercial module, a CompactCom, which communicates via SPI with an ESS in-house developed firmware driver. This paper describes the implementation of this link that was particularly challenging, as the CompactCom is designed to communicate with software entities like microprocessors, but not with FPGA firmware written in VHDL.

ANYBUS CompactCom OVERVIEW

The Anybus CompactCom module is a flexible and cheap way to connect to a PROFINET network. It is already well known and used also in the domain of Machine Protection in laboratories like CERN. In order to understand more in details the following section a brief overview of Anybus CompactCom M40 is done.

As written in the datasheet [3], the Anybus CompactCom M40 for PROFINET is a complete communication module which enables your products to communicate on a PROFINET-RT or IRT network. The module supports fast communication speeds, making it suitable also for high-end industrial devices.

Anybus CompactCom provides different application interfaces to the host: parallel, SPI, with a baudrate up to 20 MHz, shift register interface and UART. At ESS the SPI interface has been chosen as it is faster than UART and shift register interfaces. Also, even if it is slower than the parallel interface, it is less IO consuming. Then the communication is master/slave mode based, where the master, the host, is the FPGA and the slave is the Anybus CompactCom.

Figure 1, shows the interfaces available between an host and the CompactCom with details on its internal structure. In order to interface the PROFINET Network the signals

* stephane.gabourin@ess.eu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV041

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

sent by the host have to pass through Anybus CPU, the Communication Controller and finally the Physical Interface. The data flow goes also in the opposite direction.

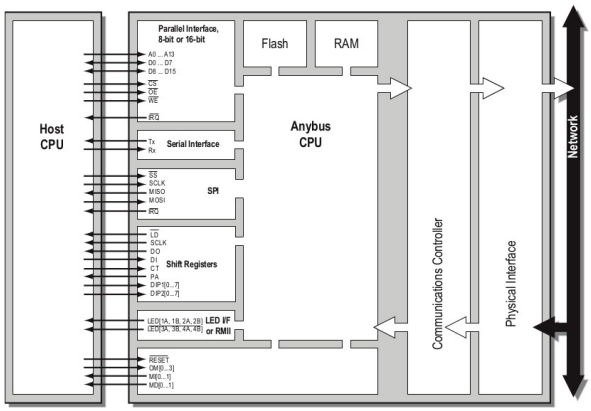


Figure 1: Host - CompactCom interface [4].

For the host the Anybus CompactCom is a grey box, where its behavior can be represented by the Anybus State machine, Fig. 2. This state machine, a Moore FSM, is a fundamental part of the Anybus CompactCom 40 that reflects the status of the module and the network. The host application is not required to keep track of all state transitions, however it is expected to perform certain tasks in each state.

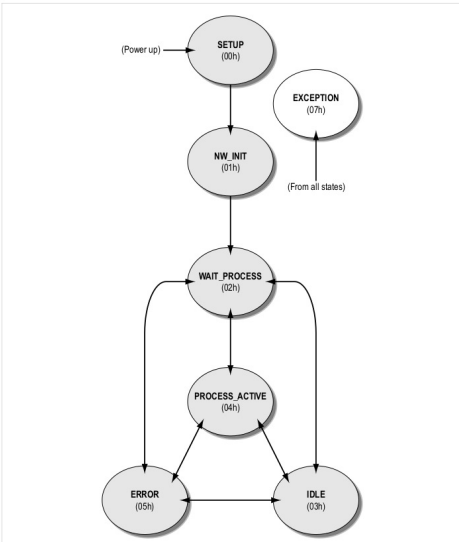


Figure 2: Anybus State Machine [4].

Lastly the host application has to be aware of the addressing scheme of the Anybus module. The software interface is object structured. According to the software documentation [4], related information and services are grouped into entities called 'Objects'. Each object can hold subentities called 'Instances', which in turn may contain a number of fields called 'Attributes'. Attributes typically represent information or settings associated with the Object. There are nine different types of objects and two are mandatory to be implemented in the host application: the Application Data Object and the Application Object.

Object messaging, between host application and Anybus module, involves two types of messages; commands and responses. On the message level, there is no master-slave relationship between the host application and the Anybus CompactCom module; both parts may issue commands, and are required to respond. Commands and responses are always associated with an instance within the Anybus object model [4].

VHDL IMPLEMENTATION

This section describes the host application code. For machine protection criticality reason, the code is written purely in VHDL and so far has been tested in two different hosts: a Xilinx Kintex UltraScale Development Kit (as shown in Fig. 3), mounting a Xilinx XCKU040 FPGA, and a custom board SCS_1600 designed by IOxOS mounting an Zynq® UltraScale+™ MPSoC XCZU7EG-FFVF1517 (as shown in Fig. 4).

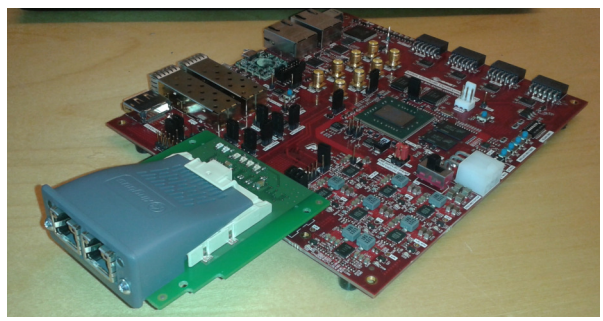


Figure 3: Anybus mounted on the Xilinx XCKU040 Development Kit via the FMC connector.

Figure 4: Highlighted the Anybus mounted on the MC. On the left and right of the crate the two cards where the Zynq® UltraScale+™ MPSoC XCZU7EG-FFVF1517 are mounted.

As written in the previous section the application has been design to support the SPI interface. The SPI frames can have two different sizes depending on if the Anybus is or not in Setup or Network Init state. To manage this, two impure functions to build MOSI frame have been written. So far only the mandatory host objects and some of their instances

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV041

and attributes are handled in the application. They have been defined as constants used to fill the SPI frames when requested.

Then two main state machines have been designed:

- the first state machine handles detection, initialization and starting of the Anybus CompactCom module
- the second state machine (named SPI FSM) is the one tracing the anybus state machine illustrated in Fig. 2, keeping communicating with the Anybus CompactCom through command and response messages.

As mentioned above the FSM in Fig. 2 is mimicked. In each of its state a subset of the SPI FSM handles communication until the Anybus reaches the Process Active state in which the system becomes fully operational. The Fig. 5 shows a simplified view of the SPI FSM subset used in Process Active. This cyclic communication was needed to:

- keep the Write Process Data updated since this data is buffered by the Anybus CompactCom, and may be sent to the network after a state shift, as recommended in [4].
- catch the responses received by the Anybus CompactCom and according to the object, instance and attribute read, build the proper following command. In the case the firmware cannot catch the response an object error is sent back.
- send the commands. Depending on the type of commands one or more SPI frames have to be delivered to the Anybus CompactCom.

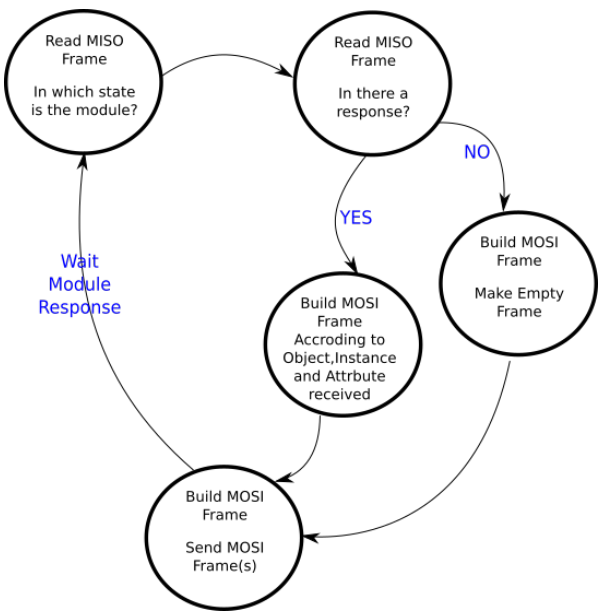


Figure 5: Detail of a status implementation in the FSM written.

With the SPI working at 20MHz and with a Process Data field of 46 bytes, the SPI frames are around 20us long, but in the process of the state machine, the data refresh rate is 1.3 ms.

FBIS is a redundant system with two channels reading data from the Anybus CompactCom. According to the SPI protocol, the Anybus CompactCom is the slave that has to be accessed by two masters SCS_1600. Currently one of the two masters engages the Anybus CompactCom, and informs the second master that stays in idle state. The second master gets then the decoded frame of the Anybus CompactCom from the first master. The link between the two masters is done via backplane and is based on a UART protocol. If the first master can't engage the module, the second master takes over, re-initializing the module and trying to reach the process active state. In case the second master fails also, the first master tries again and so on.

CURRENT STATE

The firmware has been generated using Xilinx Vivado 2018.3. As mentioned in the previous section so far it has been hardware validated in two different Xilinx device families. The maximum clock frequency tested was 250 MHz. In Table 1 a device utilization metrics for a Xilinx XCKU040 FPGA is reported.

Table 1: Xilinx XCKU040 Utilization Metrics

LUT	LUTRAM	BRAM	DSP48	Max. Freq.
2170	21	0	0	250 MHz

Currently at ESS is in the first phase commissioning of part of the normal conducting linac. FBIS interfaces three PLC systems with a PROFINET. A more detailed description of one of this system is here [5], based on PROFINET real-time fieldbus communications protocol and Anybus CompactCom M40 module.

As mentioned above, data decoded from one Anybus CompactCom has to be read by two different masters. Actually just one master engages the module instead of the second master reads decoded data from the first master.

In Fig. 6, two OPIs providing details on the status of the FSMs implemented in the two redundant masters. The OPI on the left provides details about the master that has engaged the Anybus CompactCom. The master has properly detected and identified the Anybus CompactCom module (green leds) and the module is in "Process Active" state (blue led). The master also provides module data to the UART link on the backplane. Instead of in the OPI on the right, the master doesn't read information about the module, and its logic managing the SPI communication is in idle state (blue sys_reset led). It reads information of the first master from the UART (UART alive led).

FURTHER IMPROVEMENTS

So far only a limited number of objects are caught in the code. The ones implemented were chosen empirically,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV041

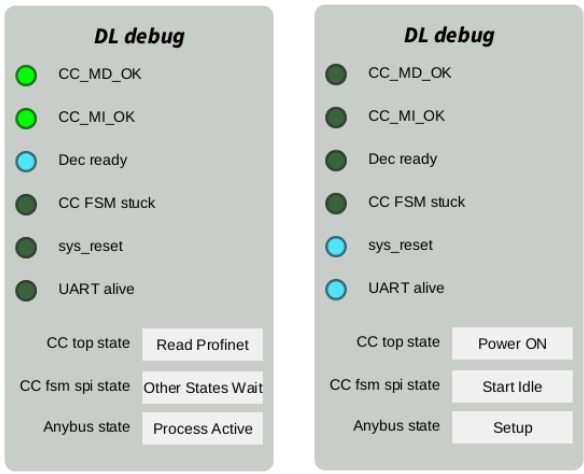


Figure 6: OPIs about the status in the two different masters. On the left the master that has engaged the Anybus CompactCom. On the right the master reading decoded data via UART.

looking at the SPI commands issued by the Anybus CompactCom. In case the module sends a command requesting an answer that is not foreseen in the host, this can lead to the Anybus state machine getting trapped in the Network Init or Wait Process state.

Now in order to overcome this, the module is reset and re-initialized. Future development will handle all possible commands and provide proper responses.

Both masters communicate through UART to arbitrate which one interfaces the module. Now it's under investigation the opportunity to exploit the backplane ethernet communication in order to replace the UART link.

The communication with the Anybus CompactCom could be managed by both master in parallel. As there is only 1 slave, the CS is whatever always in use (tied to 0) and the arbitration just has to be managed between the masters, out of the SPI protocol. However the module requests some specificity in the SPI frames that makes the dual master implementation delicate:

- a bit has to be toggled between 2 messages (command/response) except in some circumstances like errors. If it is not properly toggled, the module will not interpret messages properly.
- responses are not requested to come in the same order than commands were sent, each master has to interpret responses to check it is the one of its own command.
- some commands or responses take more than 1 SPI frame in case the Process Data field is not large enough. This adds complexity in managing the communication, especially in VHDL

CONCLUSION

An host application, written in VHDL, to interface the Anybus CompactCom has been presented.

It is integrated in the whole FBIS logic. FBIS is currently working during the first phase of the normal conducting linac for which the beam commissioning is about to start.

For the next commissioning phases we are planning to make some improvements in the code in order to facilitate the module starting, the reading during operation by our redundant system, but also to get more diagnostics from the module itself or PROFINET, and potentially send data back to PLCs.

REFERENCES

[1] R. Garoby et al., *The European Spallation Source Design* 2018 Phys. Scr. 93 (2018) 014001, doi:10.1088/1402-4896/aa9bff.

[2] T. Friedrich, C. Hilbes, and A. Nordt, "Systems of systems engineering for particle accelerator based research facilities: A case study on engineering machine protection", in *2017 Annual IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada, April 2017. doi:10.1109/SYSCON.2017.7934806

[3] HMS Industrial Networks AB, Anybus CompactCom M40 Module-PROFINET-IRT.

[4] HMS Industrial Networks AB, Anybus® CompactCom™ 40, Software Design Guide .

[5] D. Sánchez-Valdepeñas, M. Carroll, A. Nordt, and M. Zaera-Sanz, "Implementation of the PLC based Machine Protection System for Magnets at ESS", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 554–557. doi:10.18429/JACoW-ICALEPCS2019-MOPHA139

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV042

APPLYING MODEL CHECKING TO HIGHLY-CONFIGURABLE SAFETY CRITICAL SOFTWARE: THE SPS-PPS PLC PROGRAM

B. Fernández*, I. D. Lopez-Miguel, J-C. Tournier, E. Blanco,
T. Ladzinski, F. Havart, CERN, Geneva, Switzerland

Abstract

An important aspect of many particle accelerators is the constant evolution and frequent configuration changes that are needed to perform the experiments they are designed for.

This often leads to the design of configurable software that can absorb these changes and perform the required control and protection actions. This design strategy minimizes the engineering and maintenance costs, but it makes the software verification activities more challenging since safety properties must be guaranteed for any of the possible configurations.

Software model checking is a popular automated verification technique in many industries. This verification method explores all possible combinations of the system model to guarantee its compliance with certain properties or specification. This is a very appropriate technique for highly configurable software, since there is usually an enormous amount of combinations to be checked.

This paper presents how PLCverif, a CERN model checking platform, has been applied to a highly configurable Programmable Logic Controller (PLC) program, the SPS Personnel Protection System (PPS). The benefits and challenges of this verification approach are also discussed.

INTRODUCTION

Model checking is a popular verification technique that has been applied in many industries to guarantee that a critical system meets the specifications. Model checking algorithms explore all possible combinations of a system model, trying to find a violation of the formalized specification in the model. This technique is very appropriate for highly configurable projects, since it is necessary to guarantee the safety of the system for all possible configurations.

When model checking shows a discrepancy between the PLC program and the specification, it means that either the PLC program has a bug or the specification is incomplete or incorrect.

In the domain of critical PLC programs, several researchers and engineers published their experiences in the field. To name but a few, in [1] the authors translate the PLC program of an interlocking railway system, written in the FBD (Function Block Diagrams) language, into the input format language of NuSMV to verify their specification written as CTL (Computation Tree Logic) properties. In [2], the PLC program that controls the doors' opening and closing in the trains from the Metro in Brasília, Brazil, was formally verified. In this case, a B model [3] is created automatically from the PLC code. This model is formally verified using the model checker ProB [4].

When applying model checking to PLC programs, three main challenges are faced: (1) building the mathematical model of the program, (2) formalizing the requirements to be checked and (3) the state-space explosion, i.e. the number of possible input combinations and execution traces is too big to be exhaustively explored. In our case, we use the open-source tool PLCverif[5], developed at CERN. It creates automatically the models out of the PLC program and integrates three state-of-the-art model checkers: nuXmv [6], Theta [7] and CBMC [8]. It also implements some reduction and abstraction mechanisms to reduce the number of states to be explored and to speed up the verification. Therefore, challenges 1 and 3 are transparent for the user. There are certainly still limitations and large state-space PLC program models cannot be verified. Regarding challenge 2, PLCverif also provides mechanisms to help the users to formalize their requirements and provide a precise specification. However, this is normally a difficult task, specially for configurable programs. In this paper, we will show examples of functional requirements formalization with PLCverif. More details about PLCverif can be found in [5, 9].

This paper aims to show the benefits of applying model checking to verify highly configurable PLC programs. In such systems, it is usually unfeasible to check all possible combinations by traditional testing methods and model checking is a good complement to these methods, especially for module verification.

In particular, this paper shows how PLCverif was applied to the PLC programs of the SPS Personnel Protection System (PPS) [10] and how it helped to improve their original specification and correct PLC bugs before the commissioning of the system.

SPS PERSONNEL PROTECTION SYSTEM

The SPS-PPS is a large distributed control system in charge of the access control and the personnel protection of the SPS accelerator.

The SPS has 16 access zones divided in different sectors and each access zone has an access point. Several access zones are always interlocked with the same elements inhibiting operation with beam when a hazardous event is detected. This is the concept of a safety chain. Each safety chain contains the "important safety elements" to stop the beam (EISb) when a hazardous event is detected or to avoid access (EISa) when the accelerator is in Beam mode.

All details of the system can be found in [10].

* borja.fernandez.adiego@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

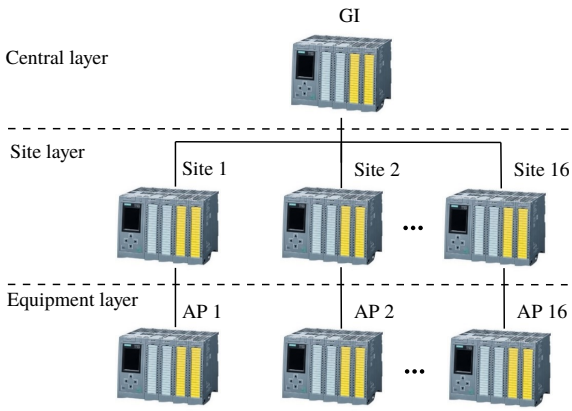


Figure 1: Simplified SPS-PPS PLC interlock architecture.

The SPS evolves with time as new access zones are added or existing ones modified. For that reason, a configurable system was designed.

Hardware Architecture

The SPS-PPS safety-interlock part is based on Siemens S7-1500F series of PLC controllers. It is formed by the following 3 layers as shown in Fig. 1:

- The central layer is composed by the Global Interlock (GI) PLC. This PLC receives the information of all the Site PLCs and computes, for example, the conditions to allow Beam or Access modes to each safety chain and the conditions to apply veto to all the EISa and EISb associated to the safety chain.
- The site layer is composed by 16 Site PLCs (1 PLC per access zone). These PLCs monitor the EISa devices (doors, emergency handles, etc.), receive the status from the AP PLCs, perform some automatic control procedures (e.g. patrol management) and apply the safety interlock actions (e.g. stop the beam) with the commands received from the GI PLC for each access zone.
- The equipment layer is composed by 16 Access Point (AP) PLCs (1 PLC per access point). They monitor and control the PAD (Personnel Access Device) and the MAD (Material Access Device) of the access point.

Software Architecture

All Site PLCs run the same generic PLC program, a common logic for all of them with a specific configuration for each access zone. The logic is composed by a set of Functions (FC) and Function Blocks (FBs) executed sequentially from a cyclic interrupt every 300 ms. Each access zone has a different configuration, indicating for example how many EISa and EISb are installed in each of them, in which sector they are located and to which safety chain they belong.

A similar approach is in place for the AP PLCs.

The GI PLC contains less configurable parameters since it communicates with all Site PLCs, having a global view of the system.

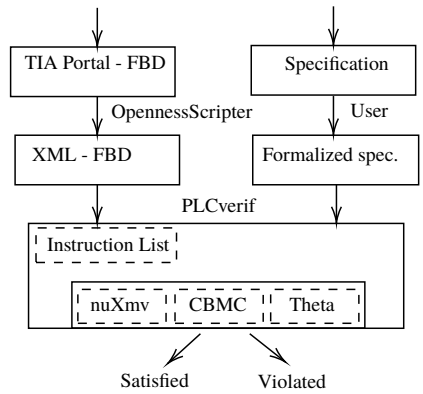


Figure 2: PLCverif workflow to verify TIA portal Safety programs.

Software Modules Specification

The different program modules (FC and FBs) are specified in a detailed design document. It contains 106 formulas that describe the requirements for the 29 FC and FBs.

The chosen formalism for the specification is a simple *if-else* conditional statement that contains Boolean formulas, as shown in the example Specification 1. The goal was to have a simple pseudo-formal specification providing precise and unambiguous requirements for each module of the system.

Specification 1 Template for specifying protection actions of the Safety program.

if (*Boolean expression condition*) **then**
  result ← 0
else if (*Boolean expression condition 2*) **then**
  result ← 1
else
  result ← *result*
end if

MODULES VERIFICATION WITH PLCVERIF

PLCverif was the tool to apply model checking to the SPS-PPS modules (FC and FBs). All the necessary model transformations to create the formal model of the PLC program are done automatically. The user only needs to provide the exported PLC program from the programming environment tool, Siemens TIA Portal[1], import it into PLCverif and finally formalize the requirements to be verified.

Figure 2 shows the workflow to verify a FC or FB from a TIA Portal safety PLC project with PLCverif:

1. The project is hosted in TIA Portal and the PLC program is written in the FBD (Function Block Diagram) programming language. An example of how this language looks like is shown in Fig. 3. Here an AND gate and an OR gate are displayed.

[1] https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software.html

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV042

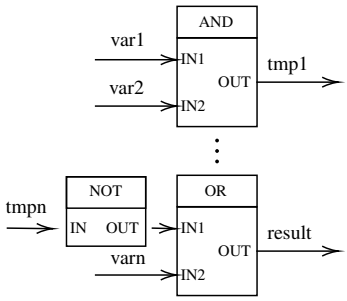


Figure 3: FBD program example.

2. Taking advantage of the Siemens OpennessScripter tool[2], the program is exported to XML (*F_FBD* XML format). An example of how an AND gate is represented in this language is shown in Listing 1.

3. PLCverif can then import that XML code, translating it into STL code, which can be understood by PLCverif. Subsequently, it creates the formal models.

4. From the specification created by the process engineers, the user needs to formalize the requirements, such as the assertion shown in Listing 2. This can be direct input to PLCverif.

5. Finally, PLCverif executes one of the model checking backends and generates a report, stating whether the property is satisfied or violated. In case it is violated, it shows the trace leading to that property failure (the counterexample).

Listing 1: XML code exported from TIA portal representing an AND gate

```
<!-- Input variables -->
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="var1"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="var2"/>
</Symbol>
</Access>

<!-- AND block -->
<Part Name="And" UId="97">
<TemplateValue Name="Card" Type="Cardinality">2</
    TemplateValue>
<TemplateValue Name="SrcType" Type="Type">Word</
    TemplateValue>
</Part>

<!-- Connecting the inputs with the block -->
<Wire UId="134">
<IdentCon UId="21" />
<NameCon UId="97" Name="in1" />
</Wire>
<Wire UId="135">
<IdentCon UId="22" />
<NameCon UId="97" Name="in2" />
</Wire>

<!-- Output variable -->
<Wire UId="136">
<NameCon UId="97" Name="out" />
<IdentCon UId="23" />
</Wire>
```

[2] https://support.industry.siemens.com/cs/us/en/view/109742322

Listing 2: Example of a PLCverif assertion

```
//#ASSERT (var1 AND var2) = result;
```

VERIFICATION RESULTS AND ANALYSIS

Although various safety functions were verified during this project, due to space limitations for this paper, only two of them will be shown here: the *SIF-X1* Function from the Site PLCs and the *SIF-2* Function from the GI PLC. Each of these PLC Functions implements several requirements of a similar nature and each of these requirements was verified with PLCverif by including an assertion which represents the mathematical Boolean formula given for that requirement. After executing a verification case with PLCverif, the following outcomes can occur:

- Satisfied property. It will be shown in Example 1 from SIF-X1.
- Violated property. The reasons why this happens need to be further investigated, leading to the following two possibilities:
  - Incomplete specification. Example 2 from SIF-2.
  - Bug in the program. Example 3 from SIF-2.

SIF-X1 Function

This function is in charge of monitoring all the EISa of the access zone and computing the status (safe or unsafe) for each safety chain.

Example 1 - Satisfied property. The selected requirement for this use case is the following: For each Safety Chain, this Function monitors all the EISa that are installed in the access zone and belong to the Safety Chain. It also receives the status from its access point (PAD, MAD and token distributor status). If all these elements are in a safe state, it returns a Boolean variable indicating that the Safety Chain is safe.

The formalized requirement is shown in Specification 2, where

i is the EISa index,

j is the safety chain index,

N_i is the number of EISa assigned to the safety chain j (the maximum number of EISa per access zone is 32),

$I_EISa_Pos[i]_Stat$ represents the status of the EISa i: true if i is closed,

$I_EISa_PU[i]_Stat$ represents the status of the emergency handle of the EISa i: true if the emergency handle is armed, $I_EISa_Pos[i]$ is a configuration variable: true if the EISa i is installed in the access zone,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

$I_EISa_PU[i]$ is configuration variable: true if the EISa i has an emergency handle,

$SC - Sj[i]$ is configuration variable: true if the EISa i belongs to the Safety Chain j,

$S0_AP_Pos[j]$, $S0_AP_PU[j]$ and $S0_AP_Key_Distrib[j]$ represent the status of the access point: true if the access point is closed, the emergency handle is armed and the token distributor is safe respectively,

$N_EISa_Safe[j]$ is the output variable of the Function and represents the status of the safety chain j: true if j is safe.

Specification 2 Partial, simplified requirement of SIF-X1.

if (

$$\prod_{\substack{i=1 \\ \forall i \in [1,N_I]:(I_EISa_Pos[i]=1 \wedge SC-S\{j\}[i]=1}}^{N_i} (I_EISa_Pos_Stat[i] \vee I_EISa_Bypass[i])$$

$$\wedge$$

$$\prod_{\substack{i=1 \\ \forall i \in [1,N_I]:(I_EISa_PU[i]=1 \wedge SC-S\{j\}[i]=1}}^{N_i} (I_EISa_PU_Stat[i] \vee I_EISa_Bypass[i])$$

$$\wedge$$

$$S0_AP_Pos[j] \wedge S0_AP_PU[j] \wedge S0_AP_Key_Distrib[j]$$

) **then**

$N_EISa_Safe[j] \leftarrow 1$

else

$N_EISa_Safe[j] \leftarrow 0$

end if

The behaviour of this relatively complex specification is shown in Table 1 with an example. Here, 2 doors (EISa) are installed in this access zone (indexes 1 and 4 of $EISa_Pos[i]$), only one of them has an emergency handle (index 1 of $EISa_PU[i]$). One of the EISa belongs to the Safety Chain 0 (index 1 of $SC - S_0[i]$) and the other one to the Safety Chain 1 (index 4 of $SC - S_1[i]$). For each safety chain j, the resultant $N_EISa_Safe[j]$ is true if the installed doors and emergency handles are in the safe state ($EISa_Pos_Stat[i]$ and $EISa_PU_Stat[i]$).

Table 1: SIF-X1 Expected Behaviour

Source	Variable	Value
Input	$EISa_Pos_Stat$	0000 0000 0000 1001
Input	$EISa_PU_Stat$	0000 0000 0000 0001
...	...	...
Input (AP PLC)	$S0_AP_Pos$	0000 0000 0000 1001
...	...	...
Configuration	$EISa_Pos$	0000 0000 0000 1001
Configuration	$EISa_PU$	0000 0000 0000 0001
...	...	...
Configuration	$SC - S_0$	0000 0000 0000 0001
Configuration	$SC - S_1$	0000 0000 0000 1000
...	...	...
Output	N_EISa_Safe	0000 0000 0000 1001

The specification was translated into 16 verification cases (one per safety chain), in order to guarantee that this property is respected in all safety chains for all possible combinations of the input and configuration variables. The example for the safety chain 0 can be seen in Listing 3 (simplified assertion).

The corresponding PLC program has 94 WORD (16-bit variable) and 4 BOOL configuration variables, and 21 WORD and 2 BOOL input variables to implement all the requirements. This implies approximately $2^{16\cdot(94+21)+6} = 2^{1846} \approx 5.0 \cdot 10^{555}$ combinations to be checked. The equivalent STL code of the original F-FBD version is around 700 lines of code.

PLCverif was able to validate the 16 verification cases in 41 seconds. The result of all properties (assertions) was satisfied. This means that no counterexamples were found and, hence, the properties always hold true.

Listing 3: SIF-X1 formalized requirement for PLCverif.

```
//#ASSERT
(((I_EISa_Pos_Stat OR I_EISa_Bypass) AND
(SC-S_0 AND I_EISa_Pos))
= (SC-S_0 OR I_EISa_Pos)) AND
(((I_EISa_PU_Stat OR I_EISa_Bypass) AND
(SC-S_0 AND I_EISa_PU))
= (SC-S_0 OR I_EISa_PU)) AND
(S0_AP_Key_Distrib.%X0 AND
S0_AP_PU.%X0 AND S0_AP_Pos.%X0)
= N_EISa_Safe.%X0
```

SIF-2 Function

The following two examples belong to the verification of the SIF-2 Function. In this function, PLCverif was able to find several discrepancies between the PLC program and the specification and two of them are presented here.

This function contains the transitions rules between the Beam and Access modes for each safety chain j. In order to allow the transition between modes, the function checks the status of the safety chain (computed variable calculated by another GI PLC function) and the requests from the operator, via the OKP (Operator Key Panel). When all safety conditions are met, this function authorizes the change of mode. The function also contains the conditions to lock or release the keys from the OKP.

The implementation of the SIF-2 Function Block has 19 WORD and 1 BOOL input variables to implement all the requirements. This implies approximately $2^{19\cdot16+1} = 2^{305} \approx 6.5 \cdot 10^{91}$ combinations to be checked.

The equivalent STL code of the original F-FBD version is around 1200 lines of code.

Example 2 - Incomplete specification. This specification was related to the conditions to switch to Beam mode. The property was formalized with the assertion shown in Listing 4:

Listing 4: SIF-2 Beam mode activation formalized requirement for PLCverif.

```
//#ASSERT
((NOT((N_EXT_ACCE_OK AND N_NO-SAFETY_ERR) AND
((I_PB_TEST_ON AND I_Key_TEST) OR
(I_PB_Acce_ON AND I_Key_Acce)))
OR (NOT N_MODE_BEAM)) = 16#FFFF)
```

This assertion was verified with PLCverif in 93 seconds and a counterexample was found, meaning that the property was violated. The reason for this violation was a missing variable in the formula from the documentation.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV042

Example 3 - Bug in the PLC program. In this case, the conditions to release an access key from the OKP for one of the safety chains were used as specification. The formalized assertion was the one shown in Listing 5:

Listing 5: SIF-2 extractors access key release formalized requirement for PLCverif.

```
//#ASSERT
((N_SECU_NO-REQ_Down.%X15 AND N_EXT_BEAM_OK.%X15)
= (SIF-2_TAGS.O_RLS_ACCESS.%X15))
```

PLCverif was able to verify this assertion in 147 seconds. The result was not satisfied and a counterexample was provided. After analysing the counterexample and discussing with the SPS-PPS experts, it was concluded that there was a problem in the PLC program. The program was modified accordingly to include the missing variables before the commissioning of the system.

Analysis

PLCverif hides most of the complexity of applying model checking to PLC programs. Nevertheless, a few manual steps were still needed in this project. In particular to import the program in PLCverif and to create the formal properties from the pseudo-formalized specification.

We also encounter the state-space explosion problem in other functions, where the number of configurations and input variables was much bigger.

Overall, PLCverif helped to validate a few critical functions from the SPS-PPS PLC program and it was a good complement to the testing activities performed in the lab by the access control experts at CERN (e.g. the SIF-X1 function).

PLCverif was able to find discrepancies between some of the specifications and the PLC program before the commissioning of the system, sometimes due to an incomplete or erroneous specification, other times due to a bug in the program (e.g. the SIF-2 function). Due to the complexity of some of the specifications, it was sometimes very hard for experts to validate the desired behaviour. By using model checking, the experts were able to find undesirable corner cases, helping them to understand the behaviour of the system and modifying the specification when necessary.

CONCLUSIONS

This paper presented a highly configurable PLC program and the benefits and challenges of applying model checking to verify it. In general, we can conclude that model checking contributed to this project to (1) detect bugs in the PLC program before the commissioning, (2) identify deficiencies in the specification, and (3) help experts to better understand the behaviour of the program for all possible configurations.

There are still several challenges to overcome for model checking to become a common practice in the industrial automation domain. Probably the more important ones are: (1) the integration of model checking in PLC programming environments (e.g. TIA portal) to minimize the number of manual steps and (2) the improvement of verification performance with better algorithms and abstraction techniques.

In the context of the PLCverif project, the future work will focus on the following directions: (1) improve PLCverif to reduce the verification time, (2) support more PLC manufacturers – we are currently working on supporting Schneider ST and IL programs. More information about the current and future work of PLCverif can be found in [9].

REFERENCES

[1] O. Pavlovic and H.-D. Ehrich, "Model checking plc software written in function block diagram," in *2010 Third International Conference on Software Testing, Verification and Validation*, 2010, pp. 439–448. DOI: 10.1109/ICST.2010.10.

[2] H. Moreira Barbosa, "Formal verification of PLC programs using the B Method," M.S. thesis, Universidade Federal do Rio Grande do Norte, Brazil, Oct. 2012.

[3] J.-R. Abrial, *The B-book - assigning programs to meanings*. Jan. 2005, ISBN: 978-0-521-02175-3.

[4] M. Leuschel and M. Butler, "The ProB Animator and Model Checker for B A Tool Description," in *International Symposium of Formal Methods Europe*, Nov. 2003, ISBN: 978-3-540-40828-4. DOI: 10.1007/978-3-540-45236-2_46.

[5] E. Blanco Viñuela, D. Darvas, and V. Molnár, "PLCverif Re-engineered: An Open Platform for the Formal Analysis of PLC Programs," 21. 7 p, 2019. DOI: 10.18429/JACoW-ICALEPCS2019-MOBPP01. https://gitlab.com/plcverif-oss/cern.plcverif

[6] R. Cavada *et al.*, "The nuxmv symbolic model checker," in *CAV*, 2014, pp. 334–342. https://nuxmv.fbk.eu/

[7] T. Tóth, Á. Hajdu, A. Vörös, Z. Micskei, and I. Majzik, "Theta: A framework for abstraction refinement-based model checking," in *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*, D. Stewart and G. Weissenbacher, Eds., 2017, pp. 176–179, ISBN: 978-0-9835678-7-5. DOI: 10.23919/FMCAD.2017.8102257. https://github.com/FTSRG/theta

[8] E. Clarke, D. Kroening, and F. Lerda, "A tool for checking ANSI-C programs," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, K. Jensen and A. Podelski, Eds., ser. Lecture Notes in Computer Science, vol. 2988, Springer, 2004, pp. 168–176, ISBN: 3-540-21299-X. https://www.cprover.org/cbmc/

[9] I. D. Lopez-Miguel, J.-C. Tournier, and B. Fernandez Adiego, "PLCverif: status of a formal verification tool for Programmable Logic Controller," in *18th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2021.

[10] T. Ladzinski, B. Fernández Adiego, and F. Havart, "Renovation of the SPS Personnel Protection System: A Configurable Approach," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2019, 395. 5 p. DOI: 10.18429/JACoW-ICALEPCS2019-MOPHA078. https://cds.cern.ch/record/2777797

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV044

BEAM PROFILE MEASUREMENTS AS PART OF THE SAFE AND EFFICIENT OPERATION OF THE NEW SPS BEAM DUMP SYSTEM

A. Topaloudis*, E. Bravin, S. Burger, S. Jackson, F. Maria Velotti, E. Veyrunes,
CERN, Geneva, Switzerland

Abstract

In the framework of the LHC Injectors Upgrade (LIU) project, the Super Proton Synchrotron (SPS) accelerator at CERN is undergoing a profound upgrade including a new high-energy beam dump. The new Target Internal Dump Vertical Graphite (TIDVG#5) is designed to withstand an average dumped beam power as high as 235 kW to cope with the increased intensity and brightness of the LIU beams whose energies in the SPS range from 14 to 450 GeV. Considering such highly demanding specifications, the constant monitoring of the device's status and the characteristics of the beams that are dumped to it is of utmost importance to guarantee an efficient operation with little or no limitations. While the former is ensured with several internal temperature sensors, a Beam Observation system based on a scintillating screen and a digital camera is installed to extract the profile of the beam dumped in TIDVG#5 for post mortem analysis. This paper describes the overall system that uses the BTV images to contribute to the safe and efficient operation of the SPS Beam Dump System (SBDS) and hence the accelerator.

INTRODUCTION

While the accelerators of the proton injector chain at CERN deliver beams to the Large Hadron Collider (LHC) within specification, the requirements for the upgraded High-Luminosity LHC (HL-LHC) exceed their current capabilities. The LHC Injectors Upgrade (LIU) project aims to address this by upgrading the proton injectors to deliver the high brightness beams needed by the HL-LHC [1].

In the framework of the LIU project, the SPS underwent several important upgrades including a new high-energy beam dump [2]. Such a system is meant to dispose of the circulating beam in the accelerator whenever necessary, i.e. in case of emergency, during machine developments (MD), LHC beam setup or LHC filling and after the slow-extraction process to eliminate the remnants of the beam for fixed targets (FT). In order to minimise the associated thermo-mechanical stresses in the dump, the energy density deposited in it is reduced by diluting the beam with the kicker magnets, producing a sinusoidal pattern on the front of the first absorbing dump block [3]. The principle is depicted in Fig. 1 along with a simulation of the expected dilution of a Fixed Target beam.

Until now, the SBDS consisted of two internal dumps, i.e. Target Internal Dump Horizontal (TIDH) and Target Internal Dump Vertical Graphite (TIDVG#4) which used to absorb beams with energy from 14 to 28.9 GeV and between 102.2 and 450 Gev accordingly. During CERN's Long Shutdown 2 (LS2) (2019-2020), the new TIDVG#5 replaced both the aforementioned dumps in order to cope with the increased intensity and brightness of the LIU beams which are expected to produce an average dumped beam power as high as 235 kW instead of 60kW. Consequently, the new system is required to withstand all beam energies in the SPS, i.e from 14 to 450 GeV, including the previously so-called "forbidden" range, 28.9–102.2 GeV; hence, removing this limitation for the beam operation [4].

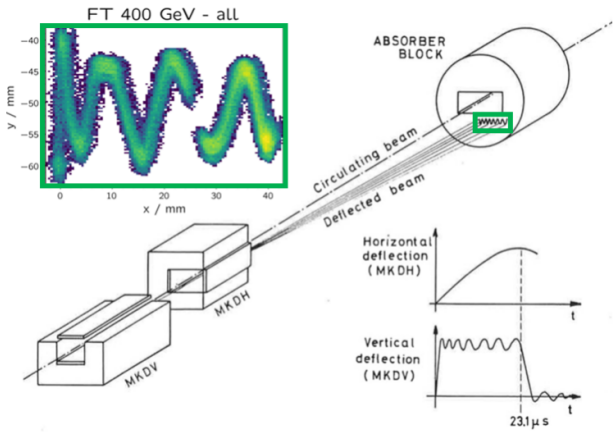


Figure 1: Principle of beam dumping in the SPS and simulation of a Fixed Target diluted beam.

MOTIVATION

In order to reduce the local energy deposition while maintaining the total required beam absorption, several innovations have been implemented in the design of the TIDVG#5, e.g. core materials, cooling and shielding. The design was done base on the most demanding beam dump scenarios taking into account possible failures of the extraction kickers responsible to dilute the particles [4].

However, to ensure the safe operation of the system and its components, the design specifications should be guaranteed. This can be achieved by constantly monitoring the characteristics of the dumped beams, i.e. the exact position of the dumped beam with respect to the dump as well as its shape and inhibit further beam injections in case of operational problems. For this, a Beam Observation system was installed to capture an image of the particles before impacting the dump block [5] to be included in the Post-Mortem analysis that verifies the quality of each dump.

* athanasios.topaloudis@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV044

BEAM OBSERVATION SYSTEM

A Beam Observation System, known at CERN as Beam TV (BTV), is widely used to acquire the image of the beam in all accelerator complexes. This is achieved by intercepting the trajectory of the beam inside the vacuum chamber with a screen. When the beam particles hit the screen, visible light is emitted proportionally to its local intensity. The resulting beam footprint can subsequently be observed by a detector e.g. a camera, through a dedicated view port and optical line [6].

SBDS Installation

In the case of the SBDS, a scintillation screen made of Alumina doped with Chromium (CHROMOX) is installed about 1.6m upstream of the dump shielding blocks [7]. The light produced on the screen is picked-up by a GigE digital camera from Basler [8] with a Complementary Metal–Oxide–Semiconductor (CMOS) sensor.

In order to protect the detector from the high levels of radiation produced on the dump block and to minimize the number of electronics Single-Event-Upset (SEU), the camera is installed in a low radiation area below the dump platform. In addition, it is contained in a dedicated shielding enclosure with a movable door to facilitate any maintenance access [7].

The detector's location was chosen following a set of FLUKA simulations [9] which also allowed the definition of the 17m optical line between the camera and the screen consisting of 5 high quality mirrors. The entire optical line volume is enclosed to avoid parasitic light and any damage on the mirrors and can be depicted in Fig. 2 [10].

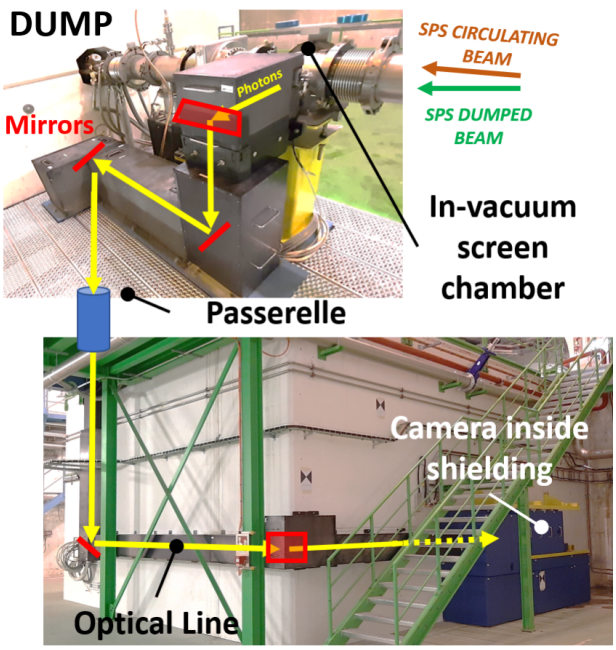


Figure 2: Optical line of the imaging system around the new SPS dump.

Acquisition System

The acquisition system at the SBDS installation is based on the relatively low decay time (> few 100 ms) of the light yield of the CHROMOX screen used. Taking advantage of the beam footprint staying on the screen for several milliseconds, the system continuously monitors the screen and selects the first image that is not saturated.

Hardware A CPU installed at a front-end computer hosts two network mezzanines, one dedicated to the communication with the camera and one for connecting the system to the control system. The camera is connected to the CPU over Ethernet through a switch and is managed by the software running in the same CPU. At the same front-end computer, a timing receiver is installed to integrate the central SPS timing to the system and custom electronics to enable the remote power management of the switch.

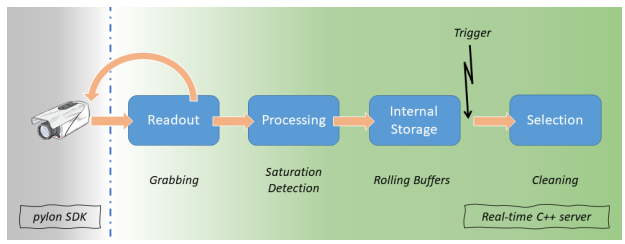


Figure 3: Image acquisition software model.

Software The real-time C++ server that orchestrates the acquisition process is designed with the Front-End Software Architecture (FESA) framework [11] and is split into two main parts:

- the *real-time* that manages the communication with the camera and its synchronization.
- the *server* that manages the communication with the clients.

The low-level access to the camera is achieved via the *pylon* Software Development Kit (SDK) [12] provided by Basler.

The image acquisition is organised in blocks that can be seen in Fig. 3. In order to improve the server's performance and avoid its blockage and consequently a potential image loss, each of the blocks run on a separate thread.

Readout During normal operation, the camera acquires images at its maximum internal frame rate, 35 Hz. The images are consecutively fed to the server by notifying the latter in the form of interrupts generated at the CPU. The server continuously grabs the new images as they arrive and copies them to intermediate storage to release the camera's buffer as soon as possible so that the camera is always ready for the next image acquisition.

Processing The copied images are subsequently provided to the analysis block for identifying if they are saturated or not. This is achieved by scanning their individual

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV044

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

pixels to find the maximum value. In a saturated image, this value will be the same as the maximum value the camera's Analogue to Digital Converter (ADC) can produce, i.e. 4095 is the maximum value that can be represented by 12 bits.

Internal Storage The images are then sent to the internal storage block to be copied to one of the software's rolling buffers. The total number of rolling buffers is configurable and by default is 8 to keep the last 7 dump events in the memory. At any given time, there is an active rolling buffer that is used only for storing the images and is not available to the user. The number of images each rolling buffer can hold is configurable up to 100, split into two categories, pre and post trigger ones.

Image Selection In order to select and publish the appropriate image, the server listens to an external trigger, i.e. that the beam was dumped, via the timing receiver. This trigger enables the selection process that counts the number of images received until the rolling buffer's size is reached, before freezing it and switching to the next available one. The first image in the rolling buffer (post-trigger) that is not saturated is selected for publication and storage in the logging and Post-Mortem data base. An example of this process is depicted in Fig. 4 with the first (saturated) image in the rolling buffer on top and the sixth (the first non saturated) in the same buffer at the bottom.

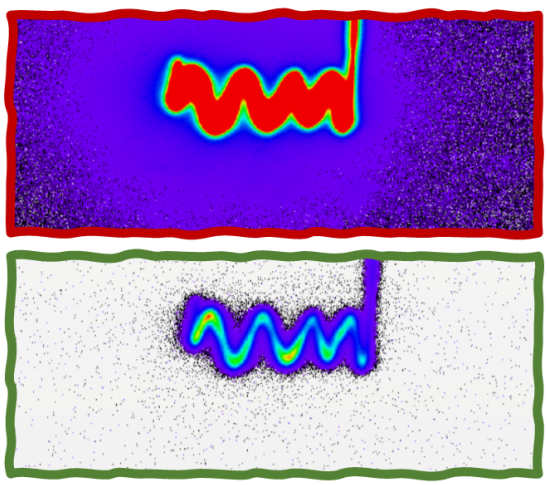


Figure 4: Example of the automatic image selection with a Fixed Target dumped beam.

Image Cleaning In order to improve the signal-to-noise ratio of the published image, an empty image, i.e. pre-trigger and without beam in the rolling buffer, is subtracted from any selected images. This process is particularly important to minimise the noise background in the published image, as light can still be present on the screen from a previous event, when two consecutive beam dumps occur relatively closely.

Image Availability Regardless of the selected image that is published, all the images of the rolling buffers, except the active one, can be retrieved before they are overwritten. This is important for verifying the aforementioned selection and cleaning mechanisms as well as the validation of the internal storage and its categorisation into pre and post trigger images.

Additionally it is also crucial during the commissioning phase of the system to set up the camera (i.e. gain, exposure time) such that it covers all SPS beam types and a non saturated image is always present in the buffers.

Internal Watchdog In order to verify the quality of each dump, the system should be 100% available so that there is always an image of the dumped beam stored for Post-Mortem analysis. The availability of the system is ensured by an internal watchdog mechanism that periodically checks and publishes the status of the acquisition and camera along with the actual image reception rate. Both statuses are subsequently monitored by the SPS Software Interlocks System [13] that inhibits further beam injections in case of anomalies.

The acquisition status is based on the image reception flagging; if they are arriving normally to the server from the camera; if there are no images arriving; or if there is an issue with their publication to the Post-Mortem database.

Moreover, the camera settings are periodically read out and checked for consistency with those expected in software. Thus the watchdog can flag if the camera status is OK, in error (i.e. having different settings) or not reachable at all.

To compliment the status publication, the mechanism includes a remote power reset of the camera via custom electronics to recover from any unexpected instability.

Operational Integration

During the SPS beam commissioning period after LS2, the Beam Observation system was validated with dumped beams varying from low intensity single bunch to trains of high intensity bunches. All features of the acquisition system were commissioned and enabled the quick setting up of the camera to cover all SPS beam types even from day 1 as they were progressively been dumped.

The system was then extensively used to commission the whole SBDS system measuring the transfer function of the kicker magnets, verifying their polarity, and assessing the effects of their potential failure.

Since the start of the SPS beam commissioning period, the system has been integrated into the operational fixed display, monitoring the SBDS status as seen in Fig. 5. The display observes various statuses of the SBDS that can be seen at the left panel. Additionally, it displays the selected non-saturated image with the dumped beam from our Beam Observation system at its center and both of the aforementioned acquisition and camera statuses on top of it. Finally, it also depicts the beam filling pattern before being dumped, at the red plot below the image.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV044

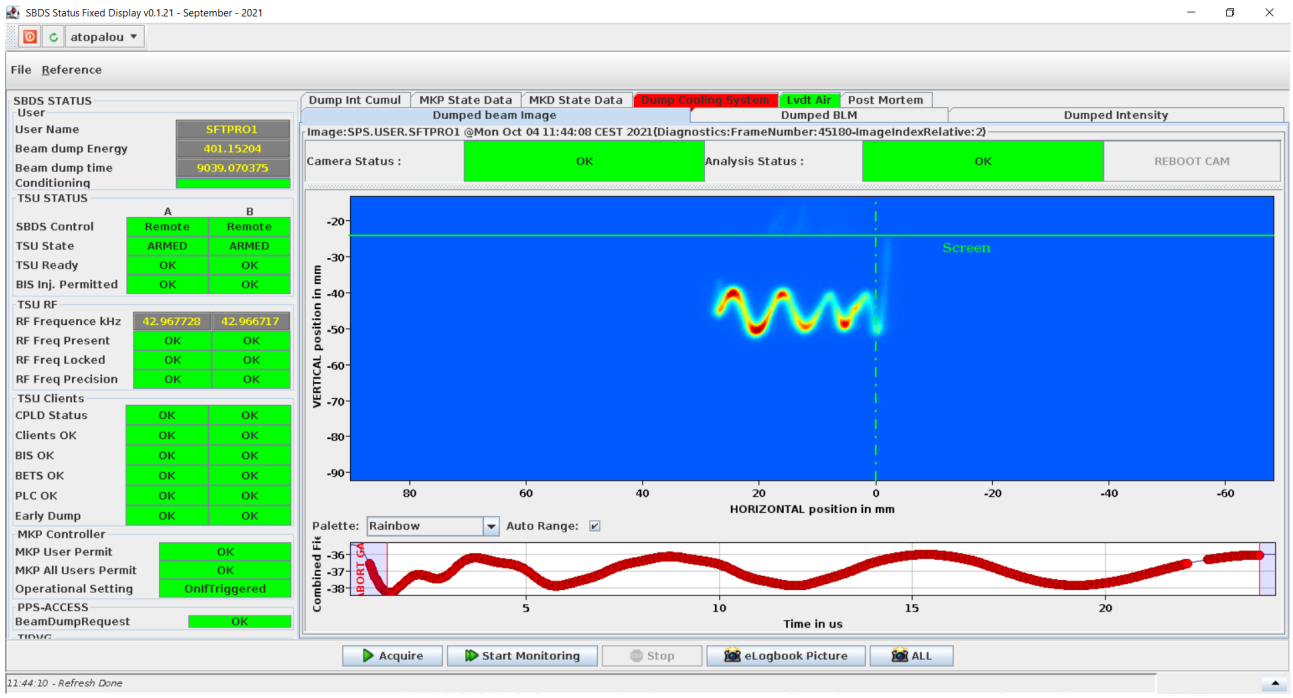


Figure 5: SBDS status Fixed Display depicting a Fix Target dump event.

CONCLUSION

In order to ensure the safe and optimal operation of the new SBDS, a Beam Observation system based on a scintillating screen and a digital camera was installed to capture and publish the image of the dumped beam.

The system acquisition is based on a free running mode, with images being acquired by the camera continuously at its maximum rate. The system integrates an *automatic detection* of the best image for publication and Post-Mortem analysis that removes the need of setting up the camera according to each beam type the SPS can produce. Additionally, it includes an *image cleaning* algorithm to improve the signal-to-noise ratio and an *internal watchdog* mechanism to monitor the statuses of the acquisition and the camera that are integrated into the SPS SIS.

During the SPS beam commissioning, the system was validated with various types of beams and was crucial in the commissioning of the new SBDS. It remains an essential part of the operational SBDS status Fix Display.

REFERENCES

[1] K. Hanke *et al.*, "The LHC Injectors Upgrade (LIU) Project at CERN: Proton Injector Chain", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 3335–3338. doi:10.18429/JACoW-IPAC2017-WEPVA036

[2] C. Pasquino *et al.*, "LHC Injectors Upgrade Project: Outlook of the Modifications to the Super Proton Synchrotron (SPS) Vacuum System and Impact on the Operation of the Carbon-Coated Vacuum Chambers", in *Proc. IPAC'18*, Vancouver, Canada, Apr.-May 2018, pp. 2589–2591. doi:10.18429/JACoW-IPAC2018-WEPMF087

[3] A. Perillo-Marcone *et al.*, "Analysis and Operational Feedback of the New High-Energy Beam Dump in the CERN SPS", in *Proc. IPAC'18*, Vancouver, Canada, Apr.-May 2018, pp. 2608–2611. doi:10.18429/JACoW-IPAC2018-WEPMG003

[4] S. Pianese *et al.*, "Design of the Future High Energy Beam Dump for the CERN SPS", in *Proc. IPAC'18*, Vancouver, Canada, Apr.-May 2018, pp. 2612–2615. doi:10.18429/JACoW-IPAC2018-WEPMG004

[5] E. Carlier, B. Goddard, A. Perillo Marcone, F. Velotti, "Beam Instrumentation Requirements for Upgraded Internal Beam Dump System in SPS LSS5", EDMS 1509780 (2021), ref SPS-TIDV-ES-0003

[6] S. Burger, E. Bravin, "A new Control System for the CERN TV Beams Observation", ref CERN-AB-Note-2008-041-BI (2008)

[7] S. Burger, L. Jensen, "BTV for SBDS beam dump diagnostics", EDMS 2008121 (2019), ref SPS-BTVD-ES-0001

[8] BASLER, https://www.baslerweb.com/en/

[9] FLUKA, http://www.fluka.org/fluka.php

[10] S. Burger, "Optical Line of the SBDS BTV system", EDMS 2060257 (2019), ref SPS-BTVD-EC-0001

[11] M. Arruat *et al.*, "Front-End Software Architecture", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.

[12] pylon Camera Software Suite, https://www.baslerweb.com/en/products/software/basler-pylon-camera-software-suite/

[13] J. Wozniak *et al.*, "Software Interlocks System", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WPPB03, pp. 403–405.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

SUPPORTING FLEXIBLE RUNTIME CONTROL AND STORAGE RING OPERATION WITH THE FAIR SETTINGS MANAGEMENT SYSTEM

R. Mueller, J. Fitzek, H. Hüther, H. Liebermann, D. Ondreka, A. Schaller, A. Walter
GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

Abstract

The FAIR Settings Management System has now been used productively for the GSI accelerator facility operating synchrotrons, storage rings, and transfer lines. The system's core is being developed in a collaboration with CERN [1], and is based on CERN's LHC Software Architecture (LSA) framework [2].

At GSI, 2018 was dedicated to integrating the Beam Scheduling System (BSS). Major implementations for storage rings were performed in 2019, while 2020 the main focus was on optimizing the performance of the overall Control System.

Integrating BSS allows us to configure the beam execution directly from the Settings Management System. Defining signals and conditions enables us to control the runtime behavior of the machine.

The Storage Ring Mode supports flexible operation with features allowing to pause the machine and execute in-cycle modifications, using concepts like breakpoints, repetitions, skipping, and manipulation.

After providing these major new features and their successful productive use, the focus was shifted on optimizing their performance. The performance was analyzed and improved based on real-word scenarios defined by operations and machine experts.

PREFACE

Patterns and Beam Production Chains (Chains) are the central technical concepts within the new LSA-based Settings Management System at GSI [3,4]. Chains are foreseen to provide a beam-oriented view on the facility from source to target, for now this is not utilized and they still represent an accelerator-oriented view. To be able to coordinate multiple beams traversing the facility in parallel, Chains are grouped into Patterns. See Fig. 1.

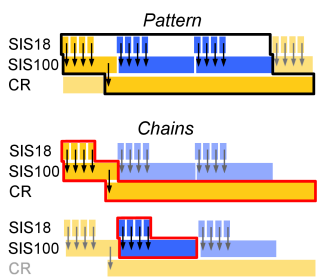


Figure 1: Patterns and Beam Production Chains as concepts for scheduling beams.

The term LSA will be used throughout this paper when the FAIR Settings Management System business logic is referenced.

RUNTIME CONTROL THROUGH BSS

Conceptually, LSA is an offline system which has no information about real-time scheduling, issued beam requests or machine status. BSS is an online system that, at runtime, processes scheduled event sequences and their dependencies on beam requests, accelerator status and beam modes.

Figure 2 shows an overview of the data that is exchanged between the systems.

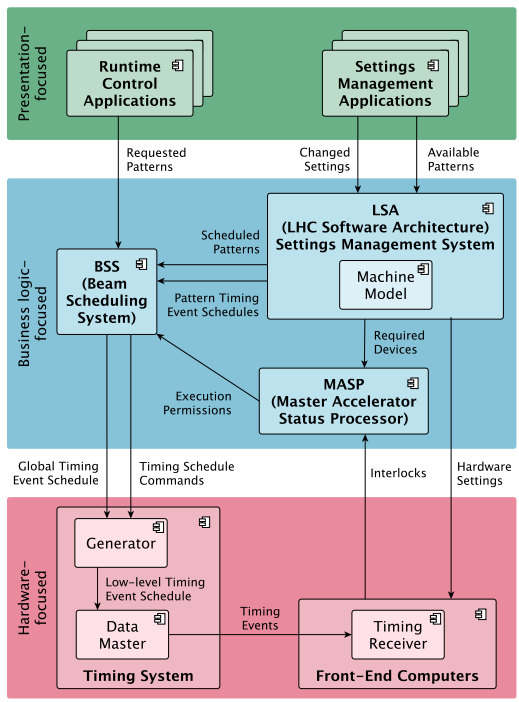


Figure 2: Control system interaction diagram.

The BSS's beam scheduling description is using LSA's own concepts like Patterns, Chains and additionally generated scheduling information like

- description of alternative timing event sequences
- definition of synchronization points
- causal description of branching depending on beam requests, accelerator status and beam-mode
- signal definitions that are used to switch between the aforementioned points

For a detailed description of BSS see [5].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

A major effort was to implement the foundation that all the information needed by BSS can be generated in LSA.

Originally at CERN, LSA only has the concept of resident contexts. A resident context represents a corresponding set of set-values, that are loaded into the devices and the Timing System could potentially send events to execute them.

At first the machine model at GSI, like the one at CERN, only contained set-values for devices. In discussion with the machine experts it was decided to introduce an additional hierarchy into the model that also calculates tables of timing events that are needed to execute the settings.

Figure 3 shows a part of the timing hierarchy that is used to calculate the timing event sequences.

LSA at GSI now has full control over the series of timing events which can be adapted depending on factors like element, isotope, energy, ramp speed/length, number of injections, etc.

The calculations lead to a series of timing events (see Table 1) that are converted into a DOT graph [6] description and sent to BSS (see Fig. 4).

Table 1: Simplified Event Sequence as Generated by the LSA Model

Timing Group	Chain	Chain Start	Event Id	Event Time µs	Comment
300	1	1	256	0	Start of sub-sequence
300	1	0	512	0	Prepare Function Generators (FG)
300	1	0	352	0	Request beam transfer
300	1	0	256	16000	Start of sub-sequence
300	1	0	351	16000	End beam transfer
300	1	0	513	16000	Start FGs and Ramp

Patterns were extended by the possibility to add execution conditions and a repetition count. Both are used by LSA to generate branching conditions. First LSA determines which branches exist and generates BSS Signals to enable switching between branches. These Signals are then used at runtime to determine the branch that should be taken for this Pattern execution.

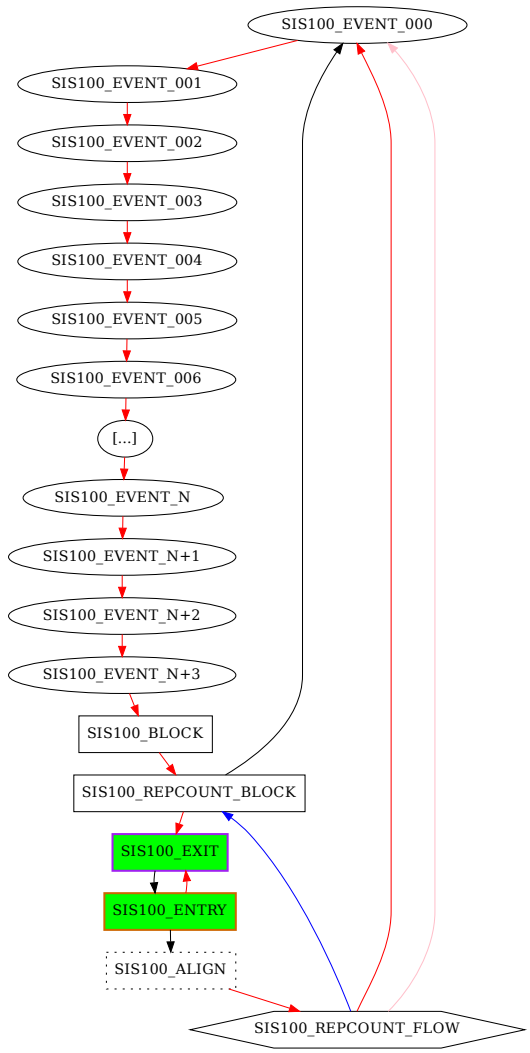


Figure 4: Schematic timing graph that is sent to BSS as DOT.

As an example Fig. 5 shows a simplified graph that allows to repeat or skip a Pattern.

Since setting manipulation happens per Pattern, the schedule graph information sent is also strictly per Pattern. Only BSS has an overview of the whole schedule.

On top of the resident concept from CERN, the concept of "Pattern Groups" has been introduced into LSA at GSI.

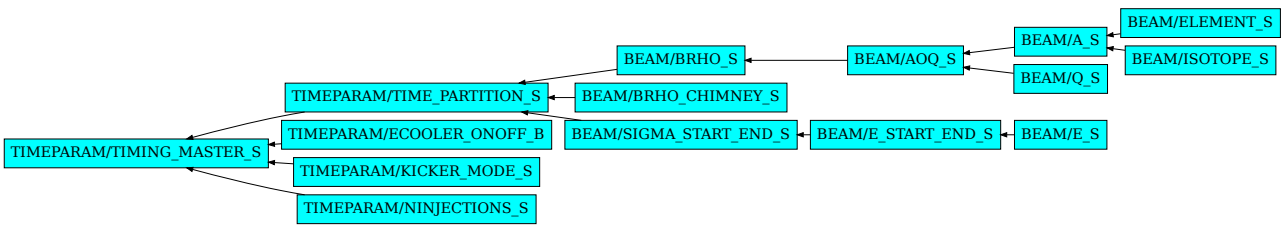


Figure 3: Simplified timing hierarchy to calculate the timing event sequence.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

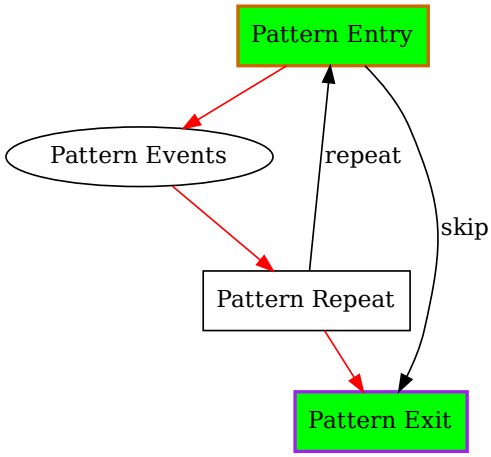


Figure 5: Simplified Pattern repetition and skip graph.

They tell BSS which Patterns can run alternately and which can be executed concurrently.

To describe that a beam is transferred between Patterns from different Pattern Groups, a concept that is called "coupling" is used. If *Pattern B* is coupled with *Pattern A*, *Pattern B* halts at injection level, requesting that *Pattern A* is executed. Once *Pattern A* is ready to transfer the beam, it signals for *Pattern B* to continue its execution. The extraction from *Pattern A* and the injection in *Pattern B* is now executed synchronously. The whole process, its timing schedule and conditions, are pre-planned offline using LSA and sent to BSS. At runtime, everything is handled at the lower-level Timing System. Figure 6 shows a simplified coupling graph.

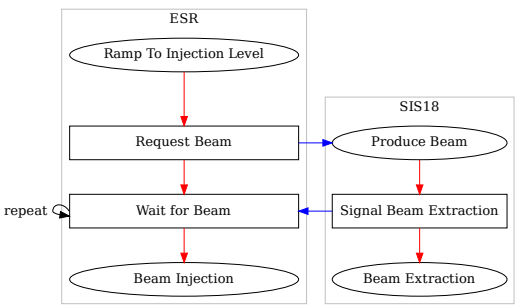


Figure 6: Schematic SIS18<->ESR coupling graph.

Realizing these features enables runtime control through BSS's signals and the timing event graph without the need of further interaction with LSA. It can be prepared beforehand what is possible at runtime.

The integration of BSS was also a precondition for Storage Ring Mode features that require manual user input at runtime to control the beam execution interactively.

STORAGE RING MODE

Ever since the commissioning of the heavy ion storage ring ESR in 1990, storage ring operation has been a regular, routine mode of operation at GSI. A second, smaller heavy ion storage ring called CRYRING [7], which had previously been installed at Stockholm University, became a part of the GSI facility in 2016.

LSA already supported fully pre-planned, usually short contexts used for synchrotron operation. As some of the storage rings can store a beam for several days, a more flexible, interactive approach that allowed certain in-cycle modifications was required.

Based on the experience of the ESR and CRYRING experts and the requirements for future storage rings at FAIR, a suitable concept for the new GSI Accelerator Control System was subsequently developed and implemented in 2019. It introduces smaller building blocks within the Chain that represents the full storage ring cycle. The desired functionality is provided by four key features for flexible operation that affect the building blocks: breakpoints, skipping, repetitions, and manipulations (see Fig. 7). Apart from manipulation, the other features can be combined with each other in a single building block.

All features can be configured and triggered by operators using a specialized Storage Ring Mode application (see Fig. 8).

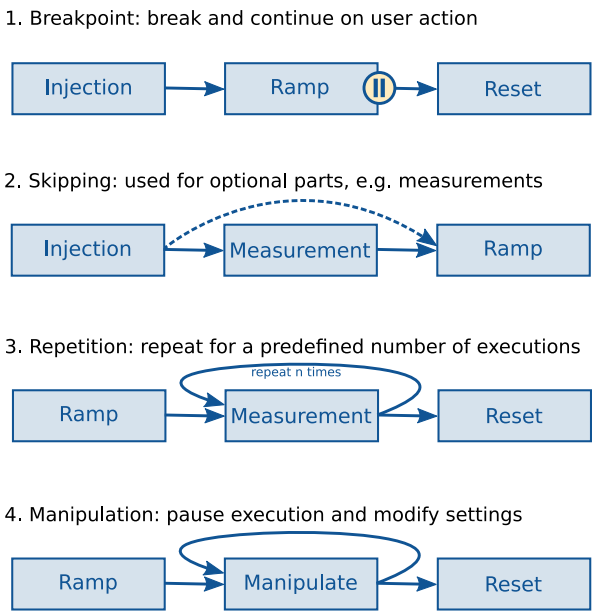


Figure 7: The four key features of the Storage Ring Mode.

Breakpoint

Breakpoints are pre-defined points at the end of a block in a storage ring Chain at which the execution can be interactively paused while the beam is still circulating in the ring. Using the specialized application, the operator can activate a breakpoint. The next time the Chain execution reaches this point, it will halt there until the operator deactivates the breakpoint and execution resumes.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

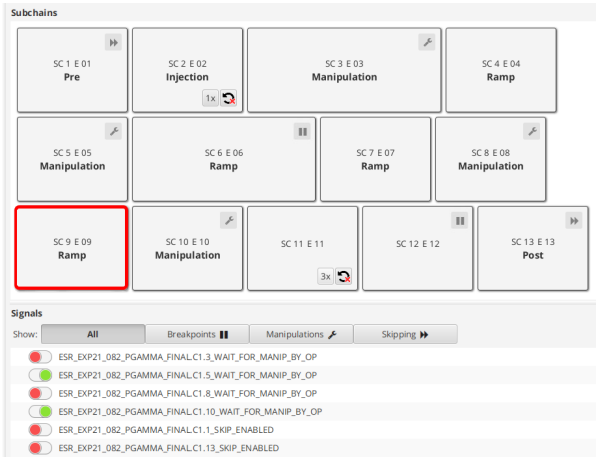


Figure 8: Partial screenshot of Storage Ring Mode application.

This feature can be used for example to pause for an indeterminate time while performing measurements, until sufficient data has been collected.

Skipping

For blocks of the storage ring Chain that are skippable, the operator can configure whether these blocks are executed or skipped.

Skipping can be used e.g. to quickly reach the end of a Chain execution and start from the beginning, for example in case of unexpected beam loss. This feature is currently also utilized to execute optional parts at the very beginning and end of a Chain. These parts bring the ring devices from their initial state up to an operational level and down again, and only need to be performed when the storage ring operation is started and stopped, but not between shots. This saves time, reduces energy consumption, and reduces the load on the devices.

Repetition

Blocks of the storage ring Chain can be repeated for a well-defined amount of repetitions. The operator can define that amount, and can decide to abort currently executed repetitions.

Repetitions are used to perform certain parts of the storage ring Chain multiple times, e.g. to take a certain number of measurements. They can also be useful to have a pre-planned way of "stalling" the execution for a pre-determined amount of time while beam is circulating – in contrast to breakpoints, which are used interactively and for an indeterminate time.

Manipulation

The most complex Storage Ring Mode feature is the manipulation. Similar to breakpoints, they offer the possibility to not only pause at certain points in the storage ring Chain, but also to modify certain settings in this state, until the operator decides to leave the paused state.

The manipulation feature is used to influence the beam while it is stored in the machine, e.g. by slightly changing its orbit. This provides the operators with a lot of flexibility to interactively find good settings by performing multiple small trims on stored beams.

Storage Ring Mode in Production

The Storage Ring Mode comprised of the features described above was successfully used in production during the beamtimes in 2020 and 2021 at ESR and CRYRING. Using this new flexibility allowed for multiple machine and science experiments.

Once Storage Ring Mode operation had been established, the performance of the Control System became the new center of attention.

PERFORMANCE OPTIMIZATION

During beamtime 2019/2020, operating identified massive performance problems with the Control System, some of which were intensified by the new features. With the system's increasing functional range and complexity, especially due to in-cycle modifications, the importance of performance as a Control System feature became apparent.

To improve the overall performance, a full stack scenario-based approach was defined [8].

Real-world scenarios that fit common use cases and are used frequently were defined, as well as very complex use cases that are not used often. A baseline measurement was performed after the beamtime for each scenario to compare the efforts that were made during shutdown 2020, see Fig. 9.

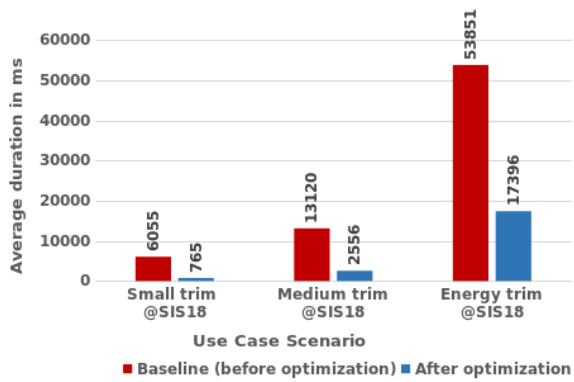


Figure 9: SIS18 scenario measurements – Total before and after optimization.

Each scenario has then been performed with applied diagnostic logging to find the parts in the stack that took a long time (see baseline in Fig. 10).

The identified long running parts were then revised by the domain experts and developers across departments working together. In addition to internal optimizations within the individual components, interfaces and APIs were redesigned to make the whole process more efficient. This full stack approach allowed precise identification of affected parts, e.g. database, BSS or LSA. In LSA, the main performance issues

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

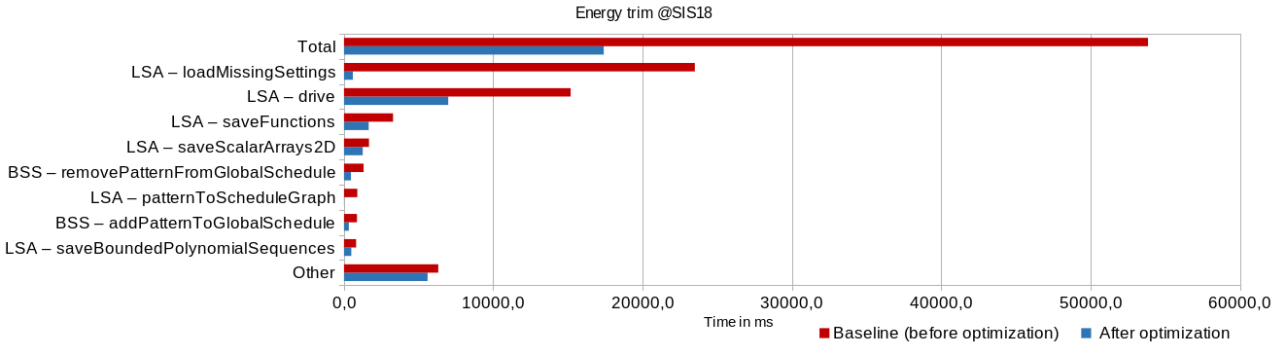


Figure 10: SIS18 scenario measurement – Critical data points before and after optimization.

were located in looking up data from and writing data to the database, and calculating new set-values. The database parts led to new and more efficient queries that were partially merged back to CERN's code base. During the trim, values were kept with higher precision than needed, leading to more complex calculations. Also more data points were kept for further calculations than needed. By reducing them to a minimum, the calculation was sped up as well as the database performance since less points have to be persisted. Also the interaction with BSS was optimized so that Patterns can be supplied more efficiently after calculating new set-values. After all optimizations had been applied through the entire Control System stack, a final measurement was made and compared to the baseline, see Fig. 10.

The overall speedup is between 7.91 for the 'Small trim' scenario down to 3.1 for the 'Energy Trim' scenario. During beamtime 2020/2021, the optimizations have been used productively and operation reported a noticeable speedup [9].

OUTLOOK

Focus is now more on features for FAIR, further discussing the general concept of "Patterns" and also reviewing the existing concepts. Other topics are requirements for upgrading the UNILAC and integrating it into the new Control System.

Current Development

Booster Mode Support Booster Mode in SIS18 is needed for FAIR. It is used for beam accumulation by stacking four SIS18 cycles into SIS100 using bunch-to-bucket transfer. At the moment, UNILAC is operated in a different way than the rest of the facility and needs to be synchronized with SIS18.

For this purpose, it was decided to introduce the possibility to start a set of timing events asynchronously in the Timing System. This way, UNILAC's deviation can be taken into account.

First tests with a first implementation that does not cover all corner cases and requires specific setup are expected during machine experiments in 2022.

Injector Controls Upgrade Project UNILAC is operated at 50 Hz, which means a cycle is only 20 ms and we can switch between cycles on this timescale. The current UNILAC Supercycle needs to be integrated into the LSA concepts, we have to make it possible to derive a current UNILAC schedule taking into account the beam requests that are expected from SIS18. Possibly the current concept of having Chains, Patterns and Pattern Groups has to be reworked.

Rework of Pattern Concept The current operating is done by executing different Patterns that each contains only one Chain while one Chain describes the beam from a source to a target. A source in this case may be an ion source or another linac or ring and a target also may be another ring or an experiment.

For the future FAIR Facility this concept needs to be developed to be more flexible. The focus will switch from machine-oriented point of view like it is now to a beam-oriented view. This means that a Pattern should contain all Chains from ion source via all participating rings and transfer lines to an experiment.

With the background of the "Injector Controls Upgrade Project" the current idea is, that there are only loosely coupled Chains within a Pattern while one Chain now describes the beam from an ion source to an experiment.

There may be multiple Patterns, but in this case, they all do different things, e.g. providing beam for an experiment on request or executing a Storage Ring Mode Chain in parallel. Ultimately, the Pattern Groups will then no longer be needed, and the Pattern itself will no longer be a context in the sense of LSA's context grouping set-values.

This allows to fulfill the special needs for the UNILAC as well as a beam-oriented view on the facility. Discussion on these concepts is ongoing.

REFERENCES

[1] R. Mueller, J. Fitzek, H. C. Hüther, and G. Kruk, "Benefits, Drawbacks and Challenges During a Collaborative Development of a Settings Management System for CERN and GSI", in *Proc. 10th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'14)*, Karlsruhe, Germany, Oct. 2014, paper TCO101, pp. 126–128.

[2] D. Jacquet, R. Gorbonosov, and G. Kruk, "LSA - the High Level Application Software of the LHC - and Its Performance

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV047

During the First Three Years of Operation", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper THPPC058, pp. 1201–1204.

[3] H. C. Hüther, J. Fitzek, R. Mueller, and A. Schaller, "Realization of a Concept for Scheduling Parallel Beams in the Settings Management System for FAIR", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 434–436. doi:10.18429/JACoW-ICALEPCS2015-MOPGF147

[4] H. C. Hüther, J. Fitzek, R. Mueller, and D. Ondreka, "Progress and Challenges during the Development of the Settings Management System for FAIR", in *Proc. 10th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'14)*, Karlsruhe, Germany, Oct. 2014, paper WPO005, pp. 40–42.

[5] S. Krepp, J. Fitzek, H. C. Hüther, R. Mueller, A. Schaller and A. Walter, "A Dynamic Beam Scheduling System for the FAIR Accelerator Facility", presented at *21th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, Shanghai, China, Oct. 2021, paper MOPV013, this conference.

[6] DOT Language, https://graphviz.org/doc/info/lang.html

[7] F. Herfurth *et al.*, "Commissioning of the Low Energy Storage Ring Facility CRYRING@ESR", in *Proc. 11th Workshop on Beam Cooling and Related Topics (COOL'17)*, Bonn, Germany, Sep. 2017, pp. 81–83. doi:10.18429/JACoW-COOL2017-THM13

[8] H. Hüther, J. Fitzek, Dr. O. Geithner, Dr. F. Herfurth, Dr. C. Hessler, Dr. S. Litvinov, Dr. B. Lorentz, S. Krepp, R. Mueller, Dr. D. Ondreka, A. Schaller, Dr. J. Stadlmann, Dr. R. Steinhagen, Dr. M. Steck, A. Walter, "Task Force Performance Project Report", *GSI annual report 2021*, GSI, Darmstadt, Germany, to be published. doi:10.15120/GSI-2021-01005

[9] J. Stadlmann, "SIS18 Report", *GSI - 4th Beam Time Retreat*, GSI, Darmstadt, Germany, 2021, not publicly available, unpublished.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

AN ARCHIVER APPLIANCE PERFORMANCE AND RESOURCES CONSUMPTION STUDY

R. Fernandes†, H. Kocevar, S. Armanet, S. Regnell, European Spallation Source, Lund, Sweden

Abstract

At the European Spallation Source (ESS), 1.6 million signals are expected to be generated by a (distributed) control layer composed of around 1 500 EPICS IOCs. A substantial amount of these signals – i.e. PVs – will be stored by the Archiving Service, a service that is currently under development at the Integrated Control System (ICS) Division. From a technical point of view, the Archiving Service is implemented using a software application called the Archiver Appliance. This application, originally developed at SLAC, records PVs as a function of time and stores these in its persistence layer. A study based on multiple simulation scenarios that model ESS (future) *modus operandi* has been conducted by ICS to understand how the Archiver Appliance performs and consumes resources (e.g. RAM memory) under disparate workloads.

INTRODUCTION

The ICS Division at ESS is mandated to deliver a system to control both its accelerator and end-station instruments. To create the system, the open-source framework EPICS [1] was chosen. With worldwide usage, EPICS allows the creation of Input/Output Controllers (IOCs) which software applications (e.g. Archiver Appliance, CS-Studio) may consume (i.e. connect to) to tackle domain specific businesses (e.g. signals archiving, signals displaying).

Typically, an IOC is an executable (i.e. software process) that utilizes resources from EPICS modules to interface (logical or physical) devices and exposes their input and output signals as Process Variables (PVs). Eventually, an IOC may also implement logic to control these devices.

A PV is a named piece of data, usually associated with devices to represent input and output signals (e.g. status, setpoint). A PV can be read, written or monitored by applications and tools using the Channel Access (CA) library.

Given that a significant number of PVs will be archived by the Archiver Appliance [2] at ESS, the present paper introduces a study to understand how this application performs when storing (i.e. writing) and retrieving (i.e. reading) PV data into and from its persistence layer thanks to a panoply of simulation scenarios designed to stress test it. In addition, the paper explores how the Archiver Appliance consumes resources (e.g. RAM memory) when handling these scenarios.

SIMULATION SCENARIOS

Thanks to discussions with domain experts to understand the type of data and volume important to test the Archiver Appliance with, four dimensions were identified along with relevant ranges of values. This helped specify simulation scenarios close to what ICS will likely face in terms of PV archiving needs and requirements from end-users, thus testing the application in a (more) meaningful way. The dimensions and ranges of values are:

- Number of PV waveforms: 1, 100, 1 000, 10 000
- Data points (per waveform): 1 000, 10 000, 100 000
- Data type: integer (4 bytes), double (8 bytes)
- Update frequency: 1 Hz, 14 Hz

Based on these, 48 simulation scenarios were specified (see Table 1).

Table 1: Simulation Scenarios

Scenario ID	Number Waveforms	Data Points	Data Type	Update Frequency
AAPS-0010	1	1 000	Integer	1
AAPS-0020	1	1 000	Integer	14
AAPS-0030	1	1 000	Double	1
AAPS-0040	1	1 000	Double	14
AAPS-0050	1	10 000	Integer	1
AAPS-0060	1	10 000	Integer	14
AAPS-0070	1	10 000	Double	1
AAPS-0080	1	10 000	Double	14
AAPS-0090	1	100 000	Integer	1
AAPS-0100	1	100 000	Integer	14
AAPS-0110	1	100 000	Double	1
AAPS-0120	1	100 000	Double	14
AAPS-0210	100	1 000	Integer	1
AAPS-0220	100	1 000	Integer	14
AAPS-0230	100	1 000	Double	1
AAPS-0240	100	1 000	Double	14
AAPS-0250	100	10 000	Integer	1
AAPS-0260	100	10 000	Integer	14
AAPS-0270	100	10 000	Double	1
AAPS-0280	100	10 000	Double	14
AAPS-0290	100	100 000	Integer	1
AAPS-0300	100	100 000	Integer	14
AAPS-0310	100	100 000	Double	1
AAPS-0320	100	100 000	Double	14
AAPS-0410	1 000	1 000	Integer	1
AAPS-0420	1 000	1 000	Integer	14
AAPS-0430	1 000	1 000	Double	1
AAPS-0440	1 000	1 000	Double	14
AAPS-0450	1 000	10 000	Integer	1
AAPS-0460	1 000	10 000	Integer	14
AAPS-0470	1 000	10 000	Double	1
AAPS-0480	1 000	10 000	Double	14
AAPS-0490	1 000	100 000	Integer	1
AAPS-0500	1 000	100 000	Integer	14
AAPS-0510	1 000	100 000	Double	1
AAPS-0520	1 000	100 000	Double	14
AAPS-0610	10 000	1 000	Integer	1
AAPS-0620	10 000	1 000	Integer	14
AAPS-0630	10 000	1 000	Double	1
AAPS-0640	10 000	1 000	Double	14

† ricardo.fernandes@ess.eu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

AAPS-0650	10 000	10 000	Integer	1
AAPS-0660	10 000	10 000	Integer	14
AAPS-0670	10 000	10 000	Double	1
AAPS-0680	10 000	10 000	Double	14
AAPS-0690	10 000	100 000	Integer	1
AAPS-0700	10 000	100 000	Integer	14
AAPS-0710	10 000	100 000	Double	1
AAPS-0720	10 000	100 000	Double	14

TOOLS

A tool named Prometheus was used to collect metrics to help evaluate the performance of the Archiver Appliance when storing and retrieving the aforementioned simulation scenarios into and from its persistence layer, as well as the stress – in terms of resources consumption – these produce on the physical machines involved in the study (i.e. MicroTCA running the IOC, Supermicro running the Archiver Appliance, and another Supermicro running a Python tool to retrieve data).

Metrics collected by Prometheus were then aggregated/displayed by another tool named Grafana. This tool can synthetize vast amounts of collected metrics very quickly in a graphical fashion through personalized dashboards – thus providing highly insightful information.

STORAGE STUDY

Environment

To enable a proper study of the Archiver Appliance in terms of PV data storage performance and resources consumption, three components were configured and used: a producer, a consumer, and a network connecting the two.

Producer The producer of PV data is an EPICS IOC running in a dedicated MicroTCA machine. The IOC was built with EPICS base version 7.0.3.1 as a 64 bit executable and uses the following EPICS modules:

- asyn (version 4.7)
- sncseq (version 2.2.8)
- procserv (version 2.8.0)
- aatest (version 1.0)
- aatestsioc (version 1.0)

In detail, the IOC is based on the EPICS asyn module and it extends the asynPortDriver C++ class. Several asyn parameters were implemented to provide control and monitoring of the PV waveforms. The IOC startup specifies how many waveforms are generated along with the number of data points for each waveform. Two types of waveforms are provided: integer (4 bytes) and double (8 bytes). On each waveform update only the first data point is modified – this is sufficient to trigger the PV update, consequently forcing the Archiver Appliance to archive the new data. It is also possible to control the delay between two consecutive waveform updates allowing the simulation of different update frequencies (e.g. 14 Hz). Multiple CPU cores may be utilized by running several IOCs at the same time, each generating/serving a fraction of the PV waveforms – crucial for running the "heaviest" simulation scenarios, which would not have been possible otherwise.

The MicroTCA machine runs CentOS 7 64 bit and has the following main characteristics:

- Manufacturer: Schroff
- Model: 3U
- CPU: Intel Xeon E3-1505M 2.80 GHz (4 cores)
- RAM: 16 GB
- Chassis: MTCA NAT-MCH

Consumer The consumer of PV data is an instance of the Archiver Appliance running in a dedicated physical storage server. The Archiver Appliance is a Java-based application that automatically records PV data in function of time in its persistence layer using Protocol Buffers [3], an extensible mechanism for serializing structured data. The instance refers to version 0.0.1 (Fall 2018 Release) of the Archiver Appliance and uses Java version 1.8.0_191 with a heap size of 16 GB. The storage server runs CentOS 7 64 bit and has the following main characteristics**:**

- Manufacturer: Supermicro
- Model: SSG-6029P-E1CR12L
- CPU: 2 x Intel 6126 2.60 GHz (12 cores per CPU)
- RAM: 128 GB
- Storage: ZFS 0.7.12 composed of 12 x 6 TB HDD (NL-SAS) and 2 x NVMe Intel P4600 4 TB

Network The network is based on a Gigabit fiber optic cable configured to transmit standard Ethernet frames (with a payload equal to 1 500 bytes) at a maximum throughput of 1 Gb/s between the producer and consumer.

Metrics

The following resources were monitored and metrics about their usages collected every five seconds while running a certain simulation scenario: CPU (load in percentage (%)), RAM (usage in gigabyte (GB)) and Network (traffic in megabit per second (Mb/s)). In addition, at the end of running the scenario, the following metrics were calculated: Disk (usage in gigabyte (GB)) and Dropped (amount of PV frames dropped in percentage (%)).

Methodology

The main method followed during the storage performance and resources consumption study of the Archiver Appliance was to have each of the 48 simulation scenarios configured to generate either 3 600 frames or 50 400 frames in a run, depending on whether the update frequency is 1 Hz or 14 Hz, respectively. This meant that each scenario would theoretically run for exactly 60 minutes.

Even though no special cache was (explicitly) implemented/configured in the MicroTCA and Supermicro machines, or in the Archiver Appliance, each simulation scenario ran with interval spaces of at least 30 minutes. This pause not only increased the confidence that the activities of a previous scenario (e.g. CPU usage) would not impact the next scenario (thus potentially distorting the collected

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

metrics) but also eased the delimitation of time range queries when subsequently displaying and analysing metrics in Grafana. Moreover, to make sure that collected metrics would not be disturbed by an unknown/undesired state of the control layer, the IOC was systematically restarted before running a new scenario.

For simulation scenarios involving 1, 100 and 1 000 waveforms, only one IOC was successful in generating these. For scenarios involving 10 000 waveforms, 8 IOCs were launched simultaneously, each generating 1 250 waveforms. The main reason for this segmentation was to spread the computation cost across multiple CPU cores (of the MicroTCA) as only one IOC trying to manage this very large number of waveforms revealed impracticable (i.e. only one IOC would saturate the CPU core where it ran and, consequently, stop responding).

To help determine the rate of data effectively stored by the Archiver Appliance, a tool was developed at ICS. This tool, implemented in Python, retrieves data stored in the Archiver Appliance and calculates the dropping factor (i.e. frames that have not been archived but should have been) of a certain PV for a given time range. Specifically, it subtracts from the number 1 the result of the division of the number of frames stored in the Archiver Appliance by the number of frames that are theoretically supposed to be stored in this application. Since the retrieval of archived data for all the PVs involved in a particular scenario revealed unfeasible (prohibitively time consuming), the solution found was to retrieve only a subset of these PVs in an evenly distributed fashion (as it was assumed that frames were being archived or not (i.e. dropped) in a normal distribution fashion – thus the dropping factor of PVs would be similar to each other).

In addition, the tool consumes a CSV-based text file containing information about each simulation scenario to know how to process it. One of the main advantages of this approach is that, in case of need, the text file can easily be extended with additional scenarios without the burden to have to modify the tool to cope with these.

Results

Table 2 contains the results of the performance and resources consumption of the Archiver Appliance from a storage point of view, while [4] provides additional details.

Table 2: Results of the Storage Study (Based on Collected Metrics)

Scenario ID	CPU Producer	RAM Producer	CPU Consumer	RAM Consumer	Disk Consumer	Network Traffic	Dropped Frames
AAPS-0010	1%	2.6 GB	< 1%	13.8 GB	< 0.1 GB	< 1 Mb/s	0%
AAPS-0020	6%	2.6 GB	< 1%	13.8 GB	0.2 GB	1 Mb/s	0%
AAPS-0030	1%	2.6 GB	< 1%	13.8 GB	< 0.1 GB	< 1 Mb/s	0%
AAPS-0040	6%	2.6 GB	< 1%	14.9 GB	0.4 GB	1 Mb/s	0%
AAPS-0050	1%	2.6 GB	< 1%	14.1 GB	0.1 GB	< 1 Mb/s	0%
AAPS-0060	10%	2.6 GB	< 1%	15.5 GB	1.9 GB	5 Mb/s	0%
AAPS-0070	1%	2.6 GB	< 1%	14.4 GB	0.3 GB	1 Mb/s	0%
AAPS-0080	10%	2.6 GB	< 1%	22.3 GB	3.8 GB	9 Mb/s	0%
AAPS-0090	1%	2.6 GB	< 1%	24.9 GB	1.3 GB	3 Mb/s	0%
AAPS-0100	10%	2.5 GB	< 1%	56.1 GB	18.7 GB	47 Mb/s	0%
AAPS-0110	1%	2.6 GB	< 1%	26.2 GB	2.7 GB	7 Mb/s	0%
AAPS-0120	10%	2.5 GB	3%	72.1 GB	37.8 GB	94 Mb/s	0%
AAPS-0210	1%	2.4 GB	< 1%	60.4 GB	1.3 GB	3 Mb/s	0%
AAPS-0220	9%	2.5 GB	< 1%	72.6 GB	18.9 GB	46 Mb/s	0%
AAPS-0230	1%	2.4 GB	< 1%	61.1 GB	2.7 GB	7 Mb/s	0%
AAPS-0240	9%	2.5 GB	2%	74.5 GB	37.8 GB	91 Mb/s	0%
AAPS-0250	1%	2.5 GB	< 1%	72.4 GB	13.4 GB	34 Mb/s	0%
AAPS-0260	9%	2.5 GB	5%	72.1 GB	188.0 GB	448 Mb/s	0%
AAPS-0270	2%	2.5 GB	1%	71.1 GB	27.0 GB	67 Mb/s	0%
AAPS-0280	11%	2.5 GB	7%	86.9 GB	350.0 GB	873 Mb/s	< 1%
AAPS-0290	2%	2.6 GB	1%	84.3 GB	138.0 GB	330 Mb/s	0%
AAPS-0300	7%	2.6 GB	6%	87.5 GB	403.0 GB	983 Mb/s	69%
AAPS-0310	3%	2.6 GB	12%	87.4 GB	242.0 GB	660 Mb/s	< 1%
AAPS-0320	9%	2.6 GB	19%	87.6 GB	418.0 GB	983 Mb/s	90%
AAPS-0410	2%	2.4 GB	< 1%	87.7 GB	13.0 GB	33 Mb/s	0%
AAPS-0420	11%	2.4 GB	1%	92.5 GB	185.0 GB	270 Mb/s	0%
AAPS-0430	2%	2.4 GB	< 1%	90.0 GB	26.0 GB	67 Mb/s	0%
AAPS-0440	12%	2.4 GB	6%	86.7 GB	370.0 GB	520 Mb/s	0%
AAPS-0450	2%	2.5 GB	1%	90.3 GB	131.0 GB	320 Mb/s	0%
AAPS-0460	3%	2.5 GB	9%	88.2 GB	263.0 GB	640 Mb/s	67%
AAPS-0470	3%	2.5 GB	9%	88.2 GB	263.0 GB	640 Mb/s	0%
AAPS-0480	10%	2.6 GB	11%	90.4 GB	517.0 GB	983 Mb/s	80%
AAPS-0490	4%	3.1 GB	7%	89.8 GB	382.0 GB	983 Mb/s	68%

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

AAPS-0500	10%	3.4 GB	11%	90.7 GB	722.0 GB	983 Mb/s	98%
AAPS-0510	4%	3.6 GB	9%	88.8 GB	134.0 GB	983 Mb/s	97%
AAPS-0520	11%	3.6 GB	6%	90.9 GB	142.0 GB	983 Mb/s	100%
AAPS-0610	10%	2.5 GB	4%	79.6 GB	140.0 GB	315 Mb/s	0%
AAPS-0620	47%	2.7 GB	9%	85.3 GB	682.0 GB	983 Mb/s	66%
AAPS-0630	8%	2.5 GB	9%	91.0 GB	250.0 GB	634 Mb/s	5%
AAPS-0640	47%	2.6 GB	4%	84.0 GB	60.0 GB	983 Mb/s	98%
AAPS-0650	7%	3.6 GB	9%	85.1 GB	340.0 GB	976 Mb/s	72%
AAPS-0660	51%	3.6 GB	12%	86.0 GB	530.0 GB	930 Mb/s	95%
AAPS-0670	7%	3.6 GB	11%	91.2 GB	260.0 GB	873 Mb/s	91%
AAPS-0680	56%	3.6 GB	10%	86.8 GB	460.0 GB	899 Mb/s	99%
AAPS-0690	11%	9.0 GB	16%	92.1 GB	210.0 GB	983 Mb/s	99%
AAPS-0700	78%	9.0 GB	11%	86.1 GB	360.0 GB	983 Mb/s	99%
AAPS-0710	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AAPS-0720	N/A	N/A	N/A	N/A	N/A	N/A	N/A

RETRIEVAL STUDY

Environment

To enable a proper study of the Archiver Appliance in terms of PV data retrieval performance and resources consumption, three components were configured and used: a producer, a consumer, and a network connecting the two.

Producer The producer of PV data is an instance of the Archiver Appliance running in a dedicated physical storage server. The instance refers to version 0.0.1 (Fall 2018 Release) of the Archiver Appliance. The storage server is a Supermicro machine and runs CentOS 7 64 bit. See subsection Environment (in section Storage Study) for additional details about the Archiver Appliance and storage server.

Consumer The consumer of PV data is a Python tool that retrieves data from the Archiver Appliance and runs in a dedicated physical machine. Thanks to its multi-threading architecture, the tool may simulate multiple clients (e.g. CS-Studio) retrieving data from the Archiver Appliance simultaneously (i.e. in parallel). The machine used for the Python tool runs CentOS 7 64 bit and has the following main characteristics:

- Manufacturer: Supermicro
- Model: SYS-1018R-WC0R
- CPU: 2 x Intel E5-2637 3.50 GHz (4 cores per CPU)
- RAM: 64 GB

Network The network is based on a Gigabit fiber optic cable configured to transmit standard Ethernet frames (with a payload equal to 1 500 bytes) at a maximum throughput of 1 Gb/s between the producer and consumer.

Metrics

The following resources were monitored and metrics about their usages collected every five seconds while running a certain simulation scenario: CPU (load in percentage (%)), RAM (usage in gigabyte (GB)) and Network (traffic in megabit per second (Mb/s)). In addition, at the end of running the scenario, the following metrics were calculated: Data Retrieved (in gigabyte (GB)) and Retrieval Time (in second (s)).

Methodology

The main method followed during the retrieval performance and resources consumption study was to retrieve PV data stored in the Archiver Appliance using a Python tool created for this purpose. It retrieved data using a RESTful interface provided by the Archiver Appliance. Although the tool was prepared to retrieve data in different formats – namely: TXT, CSV, JSON and RAW – it was decided that the retrieval study should be based on CSV since this format is the one that users will likely use when retrieving data, as well as being very suitable as a baseline/reference when compared with other formats (due to its simplicity).

Through a configuration parameter, the Python tool was able to launch multiple threads at the same time, each retrieving PV data independently from remaining threads. The idea was to simulate multiple clients (i.e. people and/or applications) retrieving data simultaneously. In this retrieval study, metrics were collected while running the tool with 1 thread, 10 threads and 100 threads.

Since the simulation scenarios based on 100, 1 000 and 10 000 PV waveforms are essentially the same as the scenarios based on 1 waveform (in terms of data points, data type and update frequency – the only difference being the number of waveforms), it was decided to only retrieve the latter scenarios from the Archiver Appliance when conducting the retrieval study.

Results

Table 3 contains the results of the performance and resources consumption of the Archiver Appliance from a retrieval point of view, while [4] provides additional details.

Table 3: Results of the Retrieval Study (Based on Collected Metrics)

Scenario ID	Number Threads	CPU Producer	RAM Producer	CPU Consumer	RAM Consumer	Network Traffic	Data Retrieved	Retrieval Time
AAPS-0010	1	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s
AAPS-0010	10	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

AAPS-0010	100	13%	71.7 GB	4%	2.0 GB	711 Mb/s	0.7 GB	5 s
AAPS-0020	1	0%	71.7 GB	1%	1.9 GB	35 Mb/s	0.1 GB	6 s
AAPS-0020	10	6%	71.7 GB	2%	2.0 GB	429 Mb/s	0.1 GB	8 s
AAPS-0020	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	9.5 GB	84 s
AAPS-0030	1	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s
AAPS-0030	10	N/A	N/A	N/A	N/A	N/A	0.1 GB	1 s
AAPS-0030	100	12%	71.7 GB	5%	2.0 GB	950 Mb/s	1.4 GB	10 s
AAPS-0040	1	1%	71.7 GB	2%	2.0 GB	69 Mb/s	0.2 GB	10 s
AAPS-0040	10	1%	71.7 GB	1%	1.9 GB	81 Mb/s	1.9 GB	16 s
AAPS-0040	100	12%	71.7 GB	5%	2.0 GB	984 Mb/s	18.9 GB	164 s
AAPS-0050	1	1%	71.7 GB	2%	1.9 GB	31 Mb/s	0.1 GB	5 s
AAPS-0050	10	4%	71.7 GB	2%	2.0 GB	302 Mb/s	0.7 GB	5 s
AAPS-0050	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	6.7 GB	57 s
AAPS-0060	1	2%	71.7 GB	2%	2.0 GB	144 Mb/s	1.0 GB	59 s
AAPS-0060	10	14%	71.7 GB	4%	2.0 GB	984 Mb/s	9.4 GB	84 s
AAPS-0060	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	94.0 GB	826 s
AAPS-0070	1	2%	71.7 GB	2%	1.9 GB	49 Mb/s	0.1 GB	7 s
AAPS-0070	10	12%	71.7 GB	4%	2.0 GB	978 Mb/s	1.3 GB	11 s
AAPS-0070	100	13%	71.7 GB	5%	2.0 GB	984 Mb/s	13.4 GB	117 s
AAPS-0080	1	2%	71.7 GB	2%	2.0 GB	158 Mb/s	1.9 GB	107 s
AAPS-0080	10	12%	71.7 GB	5%	2.0 GB	984 Mb/s	18.8 GB	169 s
AAPS-0080	100	13%	71.7 GB	5%	2.0 GB	984 Mb/s	187.9 GB	1 646 s
AAPS-0090	1	2%	71.7 GB	2%	2.0 GB	146 Mb/s	0.7 GB	41 s
AAPS-0090	10	14%	71.7 GB	5%	2.0 GB	984 Mb/s	6.7 GB	60 s
AAPS-0090	100	15%	71.7 GB	5%	2.0 GB	984 Mb/s	67.1 GB	577 s
AAPS-0100	1	2%	71.7 GB	2%	2.0 GB	142 Mb/s	9.4 GB	595 s
AAPS-0100	10	14%	71.7 GB	4%	2.0 GB	984 Mb/s	93.9 GB	840 s
AAPS-0100	100	15%	73.7 GB	5%	2.0 GB	984 Mb/s	938.9 GB	8 283 s
AAPS-0110	1	2%	73.7 GB	2%	2.0 GB	151 Mb/s	1.3 GB	79 s
AAPS-0110	10	13%	73.7 GB	5%	2.0 GB	984 Mb/s	13.4 GB	118 s
AAPS-0110	100	16%	75.8 GB	5%	2.0 GB	984 Mb/s	134.1 GB	1 153 s
AAPS-0120	1	2%	75.8 GB	2%	2.0 GB	151 Mb/s	18.8 GB	1 122 s
AAPS-0120	10	13%	75.8 GB	4%	2.0 GB	984 Mb/s	187.8 GB	1 698 s
AAPS-0120	100	16%	77.1 GB	5%	2.0 GB	984 Mb/s	1 877.7 GB	16 304 s

RESOURCES SATURATION STUDY

The results based on the metrics collected during the storage study (see Table 1) and retrieval study (see Table 2) make it evident that the network is the resource that reduced the success to store data and increased the time it took to retrieve data due to saturation. In this section, an attempt is therefore made to understand 1) the maximum number of PVs that can safely be archived (i.e. no data is dropped) by the Archiver Appliance and 2) the maximum number of threads (i.e. people and/or applications) that can concurrently retrieve data from this application if the network is never saturated (i.e. is not a bottleneck) and in function of the remaining two other resources, namely: CPU and RAM. Consequently, the attempt – which is based on extrapolations using regression lines – focuses on understanding the saturation of these two resources of the storage server. See subsection Environment (in section Storage Study) for additional details about this server. This understanding may help better manage/implement the underlying infrastructure if, for example, a system owner needs to archive a set of PVs in a proper way – by, e.g., procuring a storage server with particular characteristics – or if already deployed resources are sufficient (performance and storage wise) to cope with these PVs.

Storage

Based on the metrics collected during the PV data storage study, regression lines were calculated to model both the CPU load and the RAM usage of the storage server in function of the number of PV waveforms stored (i.e. archived) concurrently. For the calculations of these regression lines, a (single) PV waveform was assumed to have the following characteristics:

- Data points: 37 000 ((1 000 + 10 000 + 100 000) / 3)
- Data point size: 6 bytes ((4 + 8) / 2)
- Update frequency: 7.5 Hz ((1 + 14) / 2)

CPU The regression line that models the CPU load of the storage server in function of the number of PV waveforms that it may store concurrently is the following:

$$cpu_load = 0.00001 * number_waveforms + 0.02463$$

This means it takes around 100 000 waveforms being stored concurrently by the Archiver Appliance to saturate the CPU of the storage server. Above this number, the CPU becomes a bottleneck forcing the application to start dropping (i.e. not archiving) PV data.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

RAM The regression line that models the RAM usage of the storage server in function of the number of PV waveforms that it may store concurrently is the following:

$$ram_usage = 0.00003 * number_waveforms + 0.40192$$

This means it takes around 33 333 waveforms being stored concurrently by the Archiver Appliance to saturate the RAM memory of the storage server. Above this number, the RAM memory becomes a bottleneck forcing the application to start dropping (i.e. not archiving) PV data.

Retrieval

Based on the metrics collected during the PV data retrieval study, regression lines were calculated that model both the CPU load and the RAM usage of the storage server in function of the number of threads (i.e. people and/or applications) that it may serve concurrently.

CPU The regression line that models the CPU load of the storage server in function of the number of threads that it may serve concurrently is the following:

$$cpu_load = 0.00118 * number_threads + 0.02739$$

This means it takes around 847 threads to retrieve data concurrently from the Archiver Appliance to saturate the CPU of the storage server. Above this number, the CPU becomes a bottleneck forcing the application to increase the time it takes to satisfy requests.

RAM The regression line that models the RAM usage of the storage server in function of the number of threads that it may serve concurrently is the following:

$$ram_usage = 0.0001 * number_threads + 0.55964$$

This means it takes around 9 999 threads to retrieve data concurrently from the Archiver Appliance to saturate the RAM memory of the storage server. Above this number, the RAM memory becomes a bottleneck forcing the application to increase the time it takes to satisfy requests.

CONCLUSION

The results of this study imply that the Archiver Appliance appears reliable to cope with the storage (i.e. writing) of a high number of PV waveforms. This will be crucial at ESS since several thousands of devices (interfaced in EPICS) will have their control signals exposed through PV waveforms, including some of long sizes.

Thanks to the collected metrics, it can also be concluded that the Archiver Appliance consumes CPU and RAM memory in function of the number of PVs to be archived (as expected). In no simulation scenario that ran, did the CPU or the RAM of the machine running this application represent bottlenecks. Moreover, scenarios AAPS-0710 and AAPS-0720 were not executed due to the Archiver Appliance being unable to handle all the PVs (with an erratic behavior observed). Despite restarting the IOC and re-installing the Archiver Appliance, this behavior still persisted. The reason for this is unclear but we hypothesize that the sheer number of PVs and long sizes were the likely cause.

Due to the study, it is now possible to better predict/calculate if existing deployed resources (e.g. network, storage space, archiver appliance instances) are able to successfully cope with a request to, e.g., archive a new system or if additional resources are needed and provided in a preemptive fashion (thus without having to wait for PV data to be dropped to only then add more resources).

Finally, a detailed report about the performance of the Archiver Appliance and how it consumes resources to handle the simulation scenarios can be accessed in [4].

ACKNOWLEDGEMENT

The authors would like to thank everyone who helped enable the Archiver Appliance performance and resources consumption study at ESS, in particular Benjamin Bertrand, Johan Christensson, Alessio Curri, Fábio dos Santos Alves and Faye Chicken for their effective support.

REFERENCES

[1] EPICS, https://en.wikipedia.org/wiki/EPICS

[2] M. Shankar *et al*., "The EPICS Archiver Appliance", in Proc. ICALEPCS 2015, Melbourne, Australia, October 2015, paper WEPGF030.

[3] Protocol Buffers, https://en.wikipedia.org/wiki/Protocol_Buffers

[4] Archiver Appliance Performance and Resources Consumption Study, https://gitlab.esss.lu.se/ics-software/archiver-appliance-performance-and-resources-consumption-study

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV049

CONTROLS DATA ARCHIVING AT THE ISIS NEUTRON AND MUON SOURCE FOR IN-DEPTH ANALYSIS AND ML APPLICATIONS

I. D. Finch*, G. D. Howells, A. Saoulis, ISIS Neutron and Muon Source,
Rutherford Appleton Laboratory, Didcot, United Kingdom

Abstract

The ISIS Neutron and Muon Source accelerators are currently operated using Vsystem control software. Archiving of controls data is necessary for immediate fault finding, to facilitate analysis of long-term trends, and to provide training datasets for machine learning applications. While Vsystem has built-in logging and data archiving tools, in recent years we have greatly expanded the range and quantity of data archived using an open-source software stack including MQTT as a messaging system, Telegraf as a metrics collection agent, and the InfluxDB time-series database as a storage backend.

Now that ISIS has begun the transition from Vsystem to EPICS this software stack will need to be replaced or adapted. To explore the practicality of adaptation, a new Telegraf plugin allowing direct collection of EPICS data has been developed. We describe the current Vsystem-based controls data archiving solution in use at ISIS, future plans for EPICS, and our plans for the transition while maintaining continuity of data.

INTRODUCTION

The ISIS Neutron and Muon Source accelerators [1, 2] are currently controlled using the Vista Control System's software product Vsystem [3] (often colloquially called Vista), while the majority of beamlines and associated instruments [4, 5] are controlled using the Experimental Physics and Industrial Control System (EPICS) [6].

Vsystem and EPICS both originated in work done in the 1980s to create a control system for the Ground Test Accelerator Control System at Los Alamos National Laboratory [7]. While EPICS became an open-source project developed as a collaboration between multiple accelerator organisations, Vista is a closed-source commercial product with paid support. Both are distributed control systems with common features such as databases, and channels (Vsystem) or process variables (EPICS).

At ISIS Vsystem is deployed on four Itanium servers running the OpenVMS operating system. (Vsystem can be deployed on MS Windows, Linux, and OpenVMS, and operated in a hybrid configuration using any combination of these operating systems.) With the announced discontinuation of the Itanium processor architecture [8] a transition to a different processor architecture is required.

A decision has been made to transition the ISIS accelerators control system from Vsystem on OpenVMS / Itanium to EPICS on Linux / x86. This is planned as a gradual transition while ISIS is operating instead of an all-at-once conversion [9]. There will thus be a period in which both Vsystem and EPICS must operate in concert, and an early transition period in which EPICS operates only a small proportion of hardware.

This paper describes the logging functionality included with Vsystem, the software stack developed and deployed at ISIS to improve on this system, and the way in which it has been applied to machine learning and other applications at the facility. Software tools designed to allow EPICS to be used as a data source for this logging framework during the control system transition are then described.

VSYSTEM LOGGING AT ISIS

Vsystem Built-in Logging

Vsystem has an integrated logging subsystem called Vlogger [10] which consists of a Vlogger service that archives data from Vsystem databases to file, Vtrend for visualizing archived data, and utilities to manage and extract data from the generated archive files (including an SQL-like query language allowing CSV export).

At ISIS Vlogger is used to maintain short-term archives of data sampled every 30 seconds for 7 days, and longer-term data sampled every 30 minutes for 7 weeks. Both short- and longer-term archives are circular buffers. Permanent copies of the longer-term data are made at the end of each user cycle. The ISIS archive of this longer-term lower time resolution data begins in 2003, covered key channels by 2005, and subsequently steadily increased in scope.

The main limitations of this Vlogger system at ISIS are that:

- It must be run on a licensed Vsystem installation, with users therefore requiring access to and familiarity with our operations-critical OpenVMS server infrastructure.
- Channels to be logged must be specified in advance.
- It lacks a modern web-based query and visualisation interface.

For these and other reasons a parallel logging system for Vsystem was developed at ISIS.

InfluxDB Logging of Vsystem

Vsystem has an extensive and well-documented API [11] which can be used to access the Vsystem databases and their channels. Specifically, an event based callback API can be used to monitor for changes to live databases or channels.

Python code called vista_mqtt was developed to use the event based callback API to forward the changes in value and alarm states of all channels to an MQTT broker. MQTT is a publish/subscribe messaging standard and was chosen

* ivan.finch@stfc.ac.uk

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV049

for its simplicity and for its availability on OpenVMS. At the time the code was written the latest standard was 3.1.1 [12]; due to limitations in the client software there are no plans to move to MQTT 5. The MQTT broker selected was Eclipse Mosquitto [13], again chosen for its simplicity and ease of deployment. One important result of this architecture is that Mosquitto and all systems downstream (e.g. Telegraf, InfluxDB) of it may be run on the transition endpoint Linux operating system.

Vsystem as implemented at ISIS is primarily a pull-based architecture for reading, i.e. any interfaces to external systems and hardware are polled periodically. (Any writes to external systems or hardware are performed immediately.) Most polling is conducted at a 2 second cadence, though a small number of systems are polled faster or slower. The fastest cadence is 0.5 seconds for diagnostic profile and beam line monitors, some interlocks, etc. Monitored changes to values or alarm states are forwarded by vista_mqtt via MQTT immediately. In the case of reads from hardware this is at most at the polling frequency.

Telegraf [14, 15], currently run on the same Linux virtual machine as Mosquitto, subscribes to all MQTT topics published by vista_mqtt and writes the data to a single InfluxDB version 1.x time-series database [16, 17]. An upgrade to Influxdb version 2 is planned. The earliest retained data in this system dates to 2018 and the system was expanded to log changes in the value and alarm states of all single value Vsystem channels in 2019. This data is recorded at the full-time resolution. Data is interactively queried and visualised using the web-based Chronograf [18] or Grafana [19] tools.

The ISIS deployment of Vsystem currently manages more than 33,000 channels, of which approximately 1,500 may trigger alarm states. Together with the described update cadence this establishes a maximum data rate that the MQTT systems must support. In practice the data rate is significantly lower, as most values do not change at every polling interval. The current archiving rate is approximately 20 GB per user cycle (~70 days), and data rates are much lower during shutdowns.

The deployed Python-based monitoring code has proven reliable but has been found to be resource intensive on Itanium-servers which cannot be easily upgraded or expanded. Under high load this has caused system responsiveness issues and slowed down sampling of channel data which has affected the throughput of the logging system. For this reason an alternate C-based system is being developed to replicate the existing Python-based functionality, and to add monitoring of changes to other channel fields to facilitate the ongoing EPICS transition [20].

Other uses of vista_mqtt, Mosquitto, and InfluxDB

Vsystem's alarm viewer is called Valarm [21] and while reliable it is inflexible. It is a Motif-based X Window application which precludes being easily run on modern mobile devices, and as a closed source product it is difficult to integrate with locally developed software such as ISIS's FLD [22] fault diagnostics tool. Vista_mqtt publishes all changes to the alarm states of Vsystem channels and this has been used to develop a web-based alarm viewer called Rona which will address identified shortcomings.

The ability to set the value of Vsystem channel values and other fields via MQTT has been added to mqtt_vista, making it a general remote access interface to Vsystem. Although not originally designed for the purpose, this has allowed it to be leveraged to form an important component of the planned EPICS transition [9, 20].

Since the Mosquitto broker and InfluxDB time-series databases are now supported as part of normal ISIS operations, other internal groups have made use of these services. Some low-power RF functions in the synchrotron magnets are transferred via MQTT and integrated beam loss monitor and intensity data is stored in InfluxDB, though neither pass through Vsystem.

Support for Machine Learning

The relatively high time resolution logging of all inputs to the ISIS accelerators control system has been used to enable machine learning studies. One example is the real-time detection of temperature anomalies in the ISIS Target 1 methane moderator [23].

ISIS ACCELERATORS CONTROL SYSTEM TRANSITION TO EPICS

The planned transition from Vsystem to EPICS at the ISIS accelerators will be a gradual process, and initially EPICS will be a small subset of the accelerators control system. For this reason it is attractive to initially integrate EPICS into the existing logging infrastructure rather than to implement a "native" EPICS solution such as the EPICS Archiver Appliance [24]. Moving too soon to an EPICS based system would split data across two systems, making access and analysis more difficult.

In the medium-term an evaluation of EPICS logging systems will be conducted and, if necessary, a strategy will be developed to migrate our existing data holdings.

TELEGRAF AND EPICS

Telegraf is already used to aggregate and write the stream of Vsystem value changes from MQTT to InfluxDB. To support integrating EPICS into the existing logging system at ISIS, plugins have been developed for reading EPICS process variables (PVs) using Telegraf.

Telegraf is one of many metrics collection agents (others include Logstash, Metricbeat, statsd) used to collect metrics and telemetry from local and remote systems, process and aggregate that data, and then dispatch to other systems for storage or other purposes such as alert notification. Telegraf was originally selected in 2018 as it forms part of the TICK (Telegraf, InfluxDB, Chronograf, Kapacitor) stack [25] and InfluxDB had been chosen as our time-series data storage backend. In 2020 a more in-depth analysis of alternate metrics and telemetry frameworks was conducted, including a deployment of the full ELK (Elasticsearch, Logstash, Kibana) stack [26]. Our conclusion was that for our requirements there was little to

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

distinguish between the metric collection agents, and consequently little reason to switch from Telegraf given our existing expertise with the software.

Telegraf has a large number of input plugins (218 at the time of writing) used to collect data from a variety of sources and formats. EPICS is not currently among the supported data sources. There are two ways of adding the capability to handle a new source or format to Telegraf. The first is to develop a new plugin [27] for Telegraf in the Go language [28] in which Telegraf is written. The second, added in Telegraf 1.14 [29], is the use of the execd input plugin [30, 31] to collect data from an externally run program.

EPICS has two protocols used to transfer data. The Channel Access (CA) protocol [32] is the most widely deployed protocol, being supported in EPICS with revisions since at least v3.11 [33]. In EPICS v4 a new protocol called PVaccess (PVA) [34] was developed and since EPICS v7.0.1 both CA and PVA protocols have been supported by the main release series.

For the ISIS accelerator control system the intention is to use PVA where possible, and CA where necessary or convenient. This establishes a requirement to support both protocols with the existing logging software stack.

EPICS Telegraf Plugins

The CA Plugin for Telegraf developed at ISIS was based on the goca package [35]. The goca package uses Go's cgo system [36] (a binding to C libraries) to interface with the standard EPICS CA libraries. A minor modification to the package's handleEvent Go code was required to enable it to handle values of types other than double.

The CA Plugin for Telegraf is a simple development of the example Telegraf plugin and is only ~100 lines long. The channels to be monitored are simply defined in the Telegraf configuration file.

The cgo system used for the CA Plugin for Telegraf described above does not have a direct interface to C++ (or Java). There are therefore two methods available to develop a PVA plugin for Telegraf, either develop a supported interface in Go (e.g., via a shim to C) or develop a native Go implementation of the PVA protocol. Both would require significant development effort.

Instead the Telegraf execd input plugin [30] was used. This plugin allows an external program to be run as a long-running daemon and monitors the daemon's output on stdout. The execd input plugin will accept metrics in any of Telegraf's compatible input data formats [37], which Telegraf then aggregates and forwards to downstream applications.

A program called pvxs-archiver was developed in C++ using the pvxs module [38] for use with the Telegraf execd input plugin. It monitors a list of PVs listed in a configuration file and writes all changes to stdout in InfluxDB line protocol.

The CA Plugin and pvxs-archiver each handle a single EPICS protocol. Both are used to store the PVs they monitor in InfluxDB alongside any Vsystem data.

Docker and Telegraf Plugin Development

At ISIS Docker [39] containers are extensively used for development and testing [40]. Both EPICS Telegraf plugins were developed in similar Docker environments, utilising Docker Compose to instantiate containers with Telegraf running the plugin under development and InfluxDB acting as storage. Either a test IOC within a container environment or external test IOCs provided test input. This allowed a self-contained testing environment from IOC to storage backend, independent of any live systems.

Limitations

Telegraf is a metrics collection agent designed to accept timestamped metric values and timestamped log messages. It expects metrics values to be single-valued and has no direct support for more complex data structures such as 2D images. The current ISIS accelerators control system only needs to support single and 1-D array values. The latter is supported via conversion to strings which are stored in that format in InfluxDB.

CONCLUSIONS

Since 2018 the ISIS Neutron and Muon accelerators control system has greatly increased its capacity for logging Vsystem data, with a new software stack developed and deployed for this purpose. Amongst other uses the additional data collected has been used for machine learning, including detection of anomalies. As part of the transition to EPICS new plugins for Telegraf have been developed to allow designated EPICS PVs to be logged using this software stack.

REFERENCES

[1] *A Practical Guide to the ISIS Neutron and Muon Source*, Science and Technology Facilities Council, 2021, https://www.isis.stfc.ac.uk/Pages/A%20Practical%20Guide%20to%20the%20ISIS%20Neutron%20and%20Muon%20Source.pdf

[2] J. W. G. Thomason, "The ISIS Spallation Neutron and Muon Source—The first thirty-three years", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 917, pp. 61-67, February 2019. https://doi.org/10.1016/j.nima.2018.11.129

[3] Vista Control Systems, Inc., http://www.vista-control.com

[4] F. A. Akeroyd *et al.*, "IBEX - an EPICS based control system for the ISIS pulsed neutron and muon source", in *Proc. 22nd meeting of the International Collaboration on Advanced Neutron Sources (ICANS XXII)*, Oxford, United Kingdom, Mar. 2019. doi :10.1088/1742-6596/1021/1/012019

[5] K. V. L. Baker *et al.*, "IBEX: Beamline Control at ISIS Pulsed Neutron and Muon Source", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Aug. 2019. doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

[6] EPICS Control System, https://epics-controls.org

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV049

[7] L. Dalesio, A. Johnson, and K.-U. Kasemir, "The EPICS Collaboration Turns 30," in *Proc. 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Aug. 2020. doi:10.18429/JACoW-ICALEPCS2019-MOCPR02

[8] "Select Intel Itanium Processors and Intel Scalable Memory Buffer, PCN 116733-00, Product Discontinuance, End of Life", Intel, 30 January 2019. http://qdms.intel.com/dm/i.aspx/F65EEA26-13FB-4580-972B-46B75E0AB322/PCN116733-00.pdf

[9] I. D. Finch, "Evaluating Vista and EPICS with regard to future controls systems development at ISIS", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Aug. 2019. doi: 10.18429/JACoW-ICALEPCS2019-MOPHA042

[10] *Vlogger (Vsystem User's Guide V4.3)*, Los Alamos, New Mexico, Vista Controls Systems Inc., Jan. 2014.

[11] *Vaccess Reference (Vsystem User's Guide V4.3)*, Los Alamos, New Mexico, Vista Control Systems, Inc., Jan. 2014.

[12] QTT Version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[13] Eclipse Mosquitto: MQTT broker, https://mosquitto.org

[14] Telegraf: metrics agent, https://www.influxdata.com/time-series-platform/telegraf/

[15] Telegraf 1.20 documentation, https://docs.influxdata.com/telegraf/v1.20/

[16] InfluxDB: time-series database, https://www.influxdata.com/products/influxdb/

[17] InfluxDB 1.8 documentation, https://docs.influxdata.com/influxdb/v1.8/

[18] Chronograf, https://www.influxdata.com/time-series-platform/chronograf/

[19] Grafana, https://grafana.com/

[20] K. R. L. Baker, I. D. Finch, G. D. Howells, A. Saoulis, and M. Romanovschi, "pvecho: design of a Vista/EPICS bridge for the ISIS control system transition", presented at 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV019, this conference.

[21] *Valarm (Vsystem User's Guide v4.3)*, Los Alamos, New Mexico, Vista Control Systems, Inc., Jan. 2014.

[22] A. Yaqoob, J. Brower, G. Owen, R. Titmarsh, and B. Mannix, "FLD 2.2 – First Line Diagnosis", presented at the *6th Accelerator Reliability Workshop (ARW2017)*, Versailles, France, Oct. 2017.

[23] A. Saoulis, K. R. L. Baker, M. Romanovschi, S. Lilley, and R. Burridge, "Machine learning for anomaly detection in continuous signals", presented at 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper FRBL01, this conference.

[24] EPICS Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs

[25] Introduction to InfluxData's InfluxDB and TICK Stack, https://www.influxdata.com/blog/introduction-to-influxdatas-influxdb-and-tick-stack/

[26] Elastic Stack, https://www.elastic.co/what-is/elk-stack

[27] Input plugins [for Telegraf], https://github.com/influxdata/telegraf/blob/release-1.20/docs/INPUTS.md

[28] Go, https://golang.org

[29] New Plugins in v1.14 release notes, https://docs.influxdata.com/telegraf/v1.20/about_the_project/release-notes-changelog/#v114-2020-03-26

[30] execd input plugin [for Telegraf], https://github.com/influxdata/telegraf/blob/release-1.20/plugins/inputs/execd/README.md.

[31] External plugins [for Telegraf], https://docs.influxdata.com/telegraf/v1.20/external_plugins/

[32] Channel Access Protocol Specification v1.6, https://docs.epics-controls.org/en/latest/specs/ca_protocol.html

[33] EPICS Base Release v3.11, https://epics.anl.gov/base/R3-11.php

[34] pvAccess Protocol Specification, https://github.com/epics-base/pvAccessCPP/wiki/protocol

[35] M. Gibbs, goca: A Go interface to EPICS Channel Access, https://github.com/mattgibbs/goca

[36] cgo package documentation, https://pkg.go.dev/cmd/cgo

[37] Input Data Formats [for Telegraf], https://github.com/influxdata/telegraf/blob/master/docs/DATA_FORMATS_INPUT.md

[38] M. Davidsaver, PVXS, https://mdavidsaver.github.io/pvxs/overview.html

[39] Docker, https://www.docker.com/

[40] G. D. Howells and I. D. Finch, "Containerised Control Systems Development at ISIS and Potential Use in an EPICS System", presented at the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper WEPV050, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

MACHINE LEARNING TOOLS IMPROVE BESSY II OPERATION

L. Vera Ramírez*, T. Birke, G. Hartmann, R. Müller, M. Ries, A. Schälicke, P. Schnizer
Helmholtz-Zentrum Berlin, Germany

Abstract

At the Helmholtz-Zentrum Berlin (HZB), user facility BESSY II Machine Learning (ML) technologies aim at advanced analysis, automation, explainability and performance improvements for accelerator and beamline operation. The development of these tools is intertwined with improvements of the prediction part of the digital twin instances at BESSY II [1] and the integration into the Bluesky Suite [2, 3]. On the accelerator side, several use cases have recently been identified, pipelines designed and models tested. Previous studies applied Deep Reinforcement Learning (RL) to booster current and injection efficiency. RL now tackles a more demanding scenario: the mitigation of harmonic orbit perturbations induced by external civil noise sources. This paper presents methodology, design and simulation phases as well as challenges and first results. Further ML use cases under study are, among others, anomaly detection prototypes with anomaly scores for individual features.

MOTIVATION

The complexity of a large-scale facility such as the light source BESSY II in Berlin-Adlershof represents a perfect benchmark for the development, implementation and testing of Machine Learning (ML) tools due to the enormous set of use cases that can be identified - some of which were already presented in [4]. An important factor in order to prioritise these applications is the added value that might be gained through ML, which is enormous in the two cases presented in this paper.

We will first focus on a very challenging application: the mitigation of harmonic orbit perturbations. In this case ML tools aim to improve existing correction systems in the frequency domain (beyond the possibilities of current analytical methods), seeking an increase of the electron beam stability - a critical factor in order to achieve light radiation with high quality brilliance and brightness over time. Besides we will also further introduce original developments towards an anomaly detection system with feature anomaly assignation. This automatic system might extend existing preprogrammed alert system in BESSY's control room and provide additional support to human operators.

MITIGATION OF HARMONIC ORBIT PERTURBATIONS

At the light source BESSY II, the stability of the orbit in the storage ring (within the transverse beam dimensions $100 \times 20\,\mu m$) is currently pursued with a system called Fast Orbit Feedback (FOFB, [5]) running at 150 Hz. FOFB correction is based on the linear approximation $\Delta\mathbf{x} \approx S\Delta\mathbf{c}$ with $\mathbf{x}$ relative beam position, $\mathbf{c}$ corrector magnets strength and S the so-called *response matrix* (calculated or measured at the accelerator). Hence, for $\mathbf{x}_{t+1} = \mathbf{0}$ we can apply recursively $\mathbf{c}_{t+1} := \mathbf{c}_t - \alpha S^{-1}\mathbf{x}_t$ with S^{-1} Moore-Penrose pseudoinverse of S and α positive constant (at BESSY $\alpha = 0.8$).

FOFB manages to correct orbit perturbations due e.g. to imprecisions of the magnet positioning in a very efficient way. But beyond that, there are several external elements such as civil noise, main power at 50 Hz and some imperfectly isolated magnetic sources (e.g. booster power supply at 10 Hz) that also produce additional inherent perturbations (see Fig. 1). The correction induced by the FOFB system (Fig. 2) is able to mitigate perturbations at lower frequencies (less than ca. 15 Hz) but beyond that point the FOFB system is not that effective and even induces further perturbations (especially in the region 20-40 Hz).

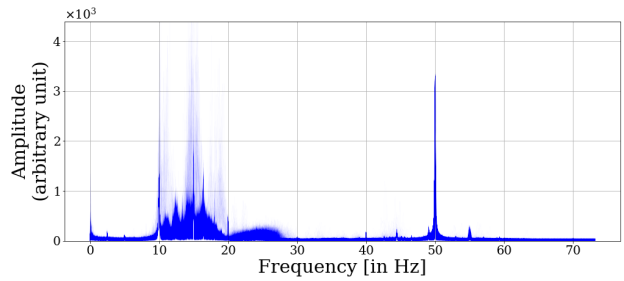


Figure 1: Horizontal beam motion spectra between injections without FOFB (cumulated along 22/04/20, BESSY Archiver data).

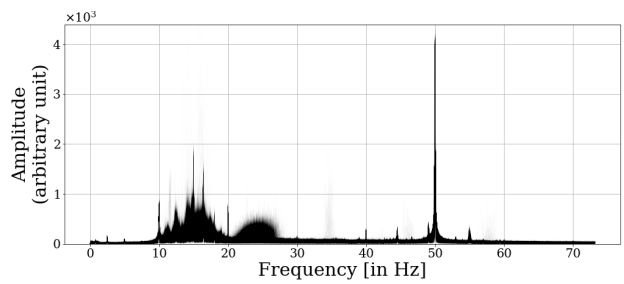


Figure 2: Horizontal beam motion spectra between injections with FOFB (cumulated along 13/05/20, BESSY Archiver data).

A first approach to face this problem was proposed, implemented and tested with simulations in [6]: an explicit correction of the 10 Hz perturbation with a *inverse wave* and an improvement of the PID correction coefficients (proportional-integral-derivative response) in the FOFB algorithm. In this work, we explore the application of ML techniques (in particular Reinforcement Learning, RL) in order to extend the analytical approach with an agent that learns the dynamics

* luis.vera_ramirez@helmholtz-berlin.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

of the system through observation and interaction directly at the machine.

Chronology

- **Early 2020**: first RL tests with simulations (code from [6] and OCELOT [7]). These feasibility tests showed that a RL agent acting in time domain with a proper configuration should be able to also reduce perturbations in frequency domain.
- **July 2020**: infrastructure set-up during machine commissioning and first plausibility tests of the Bluesky-based framework ([2]) up to 20 Hz ([3]).
- **September 2020**: direct zmq-communication with the mBox (fast orbit correction infrastructure), communication up to 100 Hz. First learning results.
- **Since February 2021**: zmq-communication improved, stable RL-interaction loop at 75 Hz, further attemps at 150 Hz.

Baselines

In all the experiments presented in this section, we will compare the performance of our models with three different baseline correction systems:

- *Static steerers*: steerers set back to the initial strength, precalculated with the slow correction system ([8]). The performance of this baseline varies due to machine drifting.
- *FOFB*: standard FOFB system for orbit correction at BESSY II ([5]). For comparison we took a 5-minute period at the end of the corresponding learning tests.
- *FOFB (zmq)*: reimplementation of the response-matrix correction (conditioning 0.04) synchronized with the zmq loop.

Model-Free Approach

The first approach that was tested in BESSY's control room was a direct translation of the simulations carried out during 2020: the orbit correction is completely undertaken by a RL agent learning and interacting in time domain in a model-free way - i.e. no explicit modelling of the environment is learnt, only the *optimisation dynamics*. According to the simulations, this agent should also be able to stabilise the orbit in frequency domain.

Following our positive experiences in [4] the chosen algorithm was Deep Deterministic Policy Gradient (DDPG, [9]). DDPG is a well-known deep-RL algorithm for continuous environments based on the update of the so-called action-value function $Q : S \times A \rightarrow \mathbb{R}$ with deterministic target policy $\mu : S \rightarrow A$, where S denotes the state space and A the action space. In the DDPG agent proposed in [9], both Q-function and policy are approximated with neural networks - a so-called *deep actor-critic* architecture. The detailed hyperparameter configuration used in our tests can be found in the Appendix.

Regarding the environment settings, we defined the **state** as the last observation of 6 beam position monitors (BPMs) along the ring measured in mm (Δ's w.r.t. the reference orbit). On the other side, we define the **action** as the strength of 6 horizontal steerers along the ring modified up to $\pm$ 20 mA (Δ's w.r.t. the initial stregth). The picked BPMs and steerers are those presenting higher absolute values in the response matrix - i.e. those presenting higher linear correlation. Finally, among the many different definitions tested for the **reward** function, we are presenting the most minimalistic and the most ambitious cases so far.

Reward defined for a single BPM: In our first tests, conceived as a proof-of-concept, the reward was defined through a exponential transformation of the deviation of the next reading from BPMZ5D7R:

$$2e^{-\beta\,|x_{t+1}|} - 1$$

with β positive constant (for this experiment $\beta = 20$). This BPM was picked because it is used as reference for the beam motion supervision at BESSY II. During this experiment the machine was set in decay mode starting at ca. 20 mA and the zmq-communication was carried out at 75 Hz.

Figure 3 shows the evolution of the beam deviation (in this case measured at a single BPM) during the whole experiment. It alternates exploration periods with baselines and exploitation periods, and the convergence can be easily perceived. Figure 4 carries out the comparison of the last baseline and exploitation period of the experiment with a subsequent FOFB running in comparable machine conditions. One can see that the agent achieves better results in time domain (for this single BPM) than any baseline.

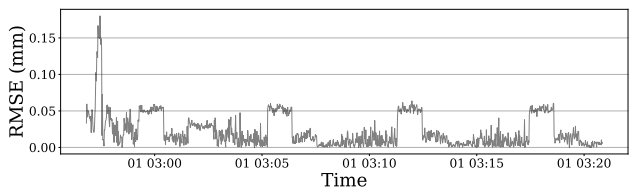


Figure 3: Beam deviation (RMSE) during the whole learning process (test with BPMZ5D7R, 1/3/21, Archiver data).

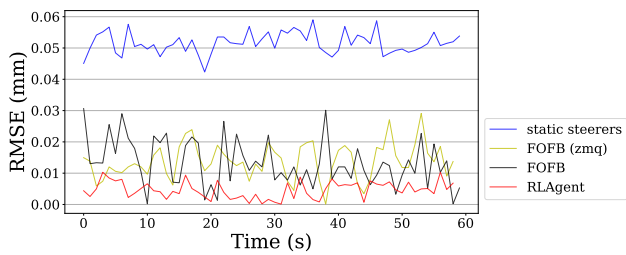


Figure 4: Comparison of the beam deviation (RMSE) after learning (test with BPMZ5D7R, 1/3/21, Archiver data).

If we switch to frequency domain with the reference beam motion measurement at BESSY II (Figs. 5 and 6), we can see that the RL agent carries out the correction of the beam position in a *smooth* way: beyond 20 Hz it improves the stability of standard FOFB and, although it does not manage to correct the inherent harmonic perturbations, it does not add any

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

extra perturbations either, as both FOFB implementations do.

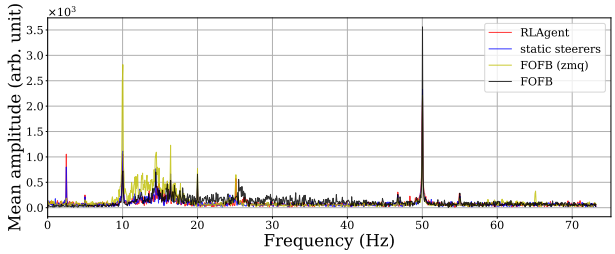


Figure 5: Mean horizontal beam motion spectra comparison (test with BPMZ5D7R, 1/3/21, Archiver data).

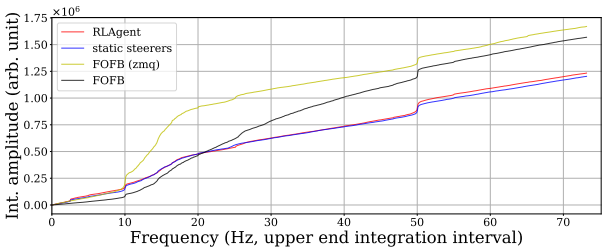


Figure 6: Integrated mean horizontal beam motion spectra comparison (test with BPMZ5D7R, 1/3/21, Archiver data).

Nevertheless, if we carry out an additional frequency analysis through the spectra of the beam motion RMSE with data gathered during the experiment (Figs. 7 and 8), we can observe a reduction of the harmonic perturbations (in norm). In this case, *FOFB* baseline can not be plotted since the data was obtained through the zmq-loop in real time and no comparable data for *FOFB* can be hence obtained.

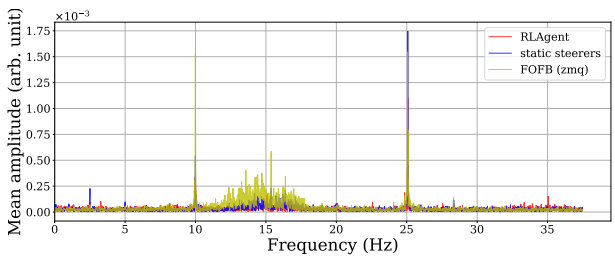


Figure 7: RMSE spectra compasion (test with BPMZ5D7R, 1/3/21, on-the-fly data).

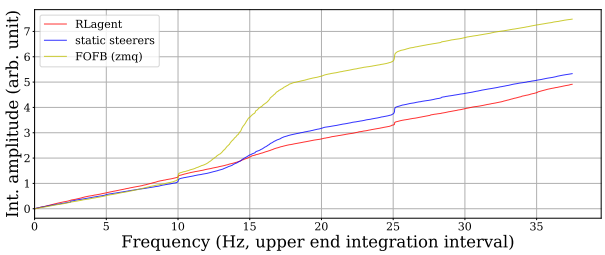


Figure 8: Integrated RMSE spectra compasion (test with BPMZ5D7R, 1/3/21, on-the-fly data).

Reward defined for all BPMs: Now we redefine the reward as an exponential transformation of the RMSE of the next BPM observation for all active BPMs (m = 102):

$$2e^{-\beta\,\mathrm{RMSE}[\mathbf{x}_{t+1}]} - 1 = 2e^{-\beta\sqrt{\frac{\sum_{i=1}^{m}\left(x_{t+1}^{i}\right)^{2}}{m}}} - 1$$

with β positive constant (here also $\beta = 20$). This is indeed a much more challenging case for the RL agent (recall that it is only observing 6 BPMs and acting on 6 steerers). In this case the machine was also set in decay mode at ca. 20 mA but the zmq-loop was carried out at 150 Hz, bringing more correction power but also some instabilities in the communication.

Figure 9 shows the evolution of the beam deviation (RMSE for all BPMs) during the whole experiment. In this period the machine suffered from drifting, which made the learning process even more difficult, affecting the baselines as well; Nevertheless, convergence can be also perceived. In the comparison showed in Fig. 10 we can see that this time the agent improves the beam deviation but does not manage to beat the FOFBs in time domain - probably the 6 steerers might not be enough for the global correction.

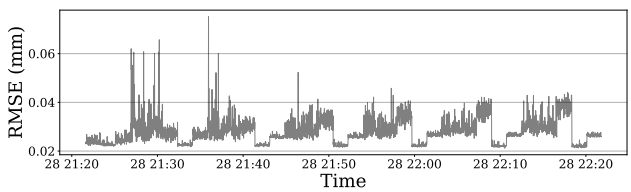


Figure 9: Beam deviation (RMSE) during the whole learning process (test with all BPMs, 1/3/21, BESSY Archiver data).

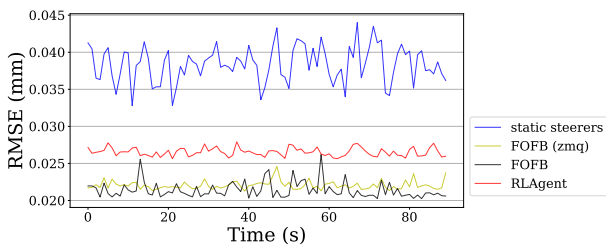


Figure 10: Comparison of the beam deviation (RMSE) after learning (test with all BPMs, 1/3/21, BESSY Archiver data).

On the other side, this experiment's outcome is more positive in frequency domain (Figs. 11 and 12): in the region between ca. 18 Hz and 50 Hz it improves the stability of all baselines, including the *static steerers* - meaning that the inherent perturbations get also slightly mitigated.

The additional frequency analysis through the spectra of the beam motion RMSE (Figs. 13 and 14) shows an even stronger reduction of the harmonic perturbations (in norm).

A natural question might be if the reason for stability in frequency domain is a *low* activity of the steerers when they are controlled by the RL agent. Figure 15 shows that it does not seem to be the case - the 6 steerers manipulated by the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

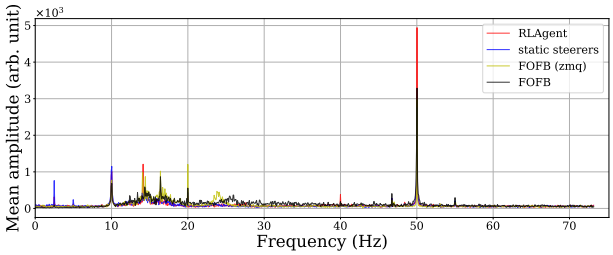


Figure 11: Mean horizontal beam motion spectra comparison (test with all BPMs, 1/3/21, BESSY Archiver data).

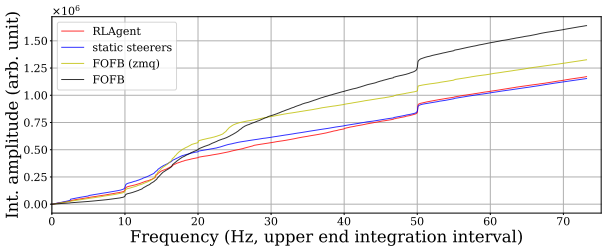


Figure 12: Integrated mean horizontal beam motion spectra comparison (test with all BPMs, 1/3/21, BESSY Archiver data).

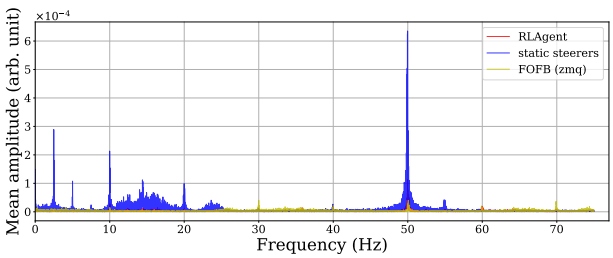


Figure 13: RMSE spectra compasion (test with all BPMs, 1/3/21, on-the-fly data).

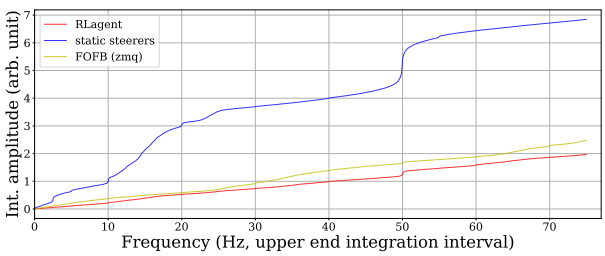


Figure 14: Integrated RMSE spectra compasion (test with all BPMs, 1/3/21, on-the-fly data).

agent present active motion patterns, which remind of the beam motion perturbations they are supposed to counteract.

Towards a Model-Based Approach

Previous experiments with a model-free RL agent were able to carry out sensible orbit corrections in frequency domain. Nevertheless, in time domain traditional algorithms still have a major advantage when the target involves the correction of all BPMs: the need to include more horizontal

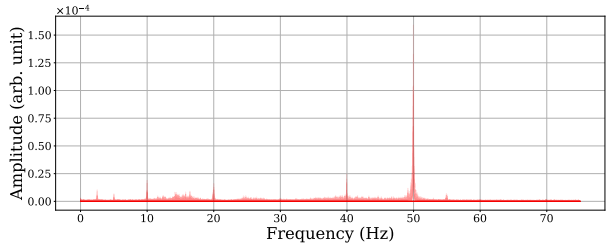


Figure 15: Steerers motion spectra with RL agent (test with all BPMs, 1/3/21, on-the-fly data).

steerers implies an increase of the action space dimensionality and complicates the convergence of the RL agent.

In this sense, an optimal set up might be a *combination* between global response-matrix-based correction "smoothed" with RL for the frequency domain. Our first attempts in this direction (RL agent action added to or introduced into the global response-matrix correction) were not successful - probably because the dynamics of the environment got more complicated to learn with the standard model-free RL loop.

As a consequence, we have started exploring some model-based approaches. The idea is to *encapsulate* the dynamic system induced by the harmonic perturbation patterns within a surrogate model that can be used for combination with the response-matrix-based correction but also for policy training in a model-based RL context. We seek hence a surrogate model in the form

$$F : \mathbb{R}^{n\times m} \times \mathbb{R}^{l} \longrightarrow \mathbb{R}^{m}$$
$$(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t, \mathbf{c}_{t+1}) \longmapsto \mathbf{x}_{t+1}$$

where n is the window size, m the number of BPMs and l the number of steerers.

In our first tests of this approach at the machine, the role of surrogate model was undertaken by a neural network whose architecture is sketched in Fig. 16. This network was directly fed with real data obtained via zmq in real-time interaction with the machine: random steerer strengths were set sporadically and the BPM response to these modifications was tracked. But since the steerer modifications were carried out slowly (see Appendix), the (windowed) BPM data also included the inherent harmonic perturbations.

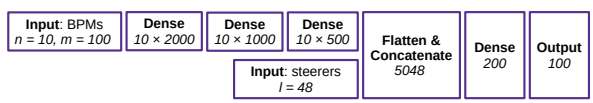


Figure 16: Surrogate model architecture schema.

Table 1 shows that system dynamics were accurately learnt by the model. In this table, model prediction error on unseen test data is compared with several baselines: average of the test data, previous observations in the test data ($\mathbf{x}_t$, $\mathbf{x}_{t-1}$,$\mathbf{x}_{t-2}$, $\mathbf{x}_{t-3}$ and $\mathbf{x}_{t-4}$) and prediction obtained by the application of the response matrix: $\mathbf{x}_{t+1} \approx \mathbf{x}_t + S(\mathbf{c}_{t+1} - \mathbf{c}_t)$.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 1: Surrogate Model and Baseline Prediction Errors

Error	Test Set Avg.	Previous BPM Data					Resp. Matrix:	Model:
		$\mathbf{x}_t$	$\mathbf{x}_{t-1}$	$\mathbf{x}_{t-2}$	$\mathbf{x}_{t-3}$	$\mathbf{x}_{t-4}$	$\mathbf{x}_t + S\,(\mathbf{c}_{t+1} - \mathbf{c}_t)$	$F(\mathbf{x}_{t-9}, \ldots, \mathbf{x}_t, \mathbf{c}_{t+1})$
RMSE	0.0154	0.0274	0.0178	0.0236	0.0204	0.0229	0.0098	0.0036
R^2	0	-2.172	-0.3455	-1.3492	-0.769	-1.2223	0.5896	0.9441

The impressive performance of the surrogate model encouraged us to embed it into an algorithm (sketched as Algorithm 1) where its prediction accuracy is used in order to recursively improve the correction given by the response matrix pseudo-inverse. In each iteration, the implicit linear prediction calculated with the response matrix is replaced by the model prediction, which takes into account the inherent orbit perturbations that are not captured by the measured response matrix.

Data: S response matrix, S^{-1} response matrix pseudoinverse, $\mathbf{x}_t$ current BPMs, $\mathbf{c}_t$ current steerer strengths (corrections), F surrogate model

$c^0_{t+1} := \mathbf{c}_t - \alpha S^{-1}\mathbf{x}_t$

repeat

$\quad \mathbf{x}^k_{t+1} := \mathbf{x}_t + S(\mathbf{c}^k_{t+1} - \mathbf{c}_t)$

$\quad \tilde{\mathbf{x}}^k_{t+1} := F(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t, \mathbf{c}^k_{t+1})$

$\quad \mathbf{c}^{k+1}_{t+1} := \mathbf{c}_t + \alpha S^{-1}\big[- (\tilde{\mathbf{x}}^k_{t+1} - \mathbf{x}^k_{t+1}) - \mathbf{x}_t\big]$

until *convergence or k too large*

Algorithm 1: Combination of surrogate model and response-matrix-based correction.

A test run at the machine of Algorithm 1 (300 mA in decay mode, 150 Hz, $k = 1$) can be visualised in Figs. 17, 18 and 19. Notice that the algorithm is directly constructed over the baseline *FOFB (zmq)*, and the results are coherent with its theoretical construction: the performance in time domain is quite similar to *FOFB (zmq)*'s, whereas it is much more stable in frequency domain.

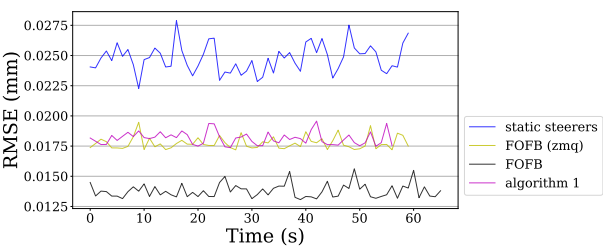


Figure 17: Comparison of the beam deviation (RMSE) after learning (test with surrogate model, 21/6/21, BESSY Archiver data).

Interpretability: A major advantage of the model-based approach we are seeking is the possibility of a more direct interpretability analysis. As an example, we want to briefly analyse the behaviour of the surrogate model F trained in the previous section. Recall that F is a neural network with ReLU and linear activation functions; it means that F is differentiable almost everywhere and with the help of Tensorflow's automatic differentiation we can approximate its *jacobian* w.r.t. the steerer strengths:

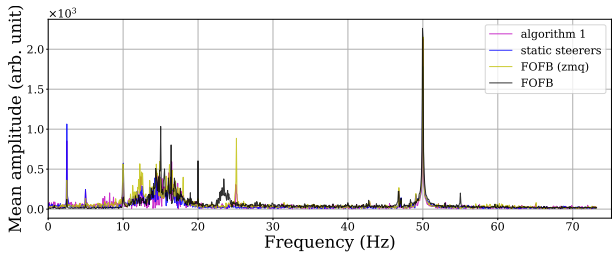


Figure 18: Mean horizontal beam motion spectra comparison (test with surrogate model, 21/6/21, BESSY Archiver data).

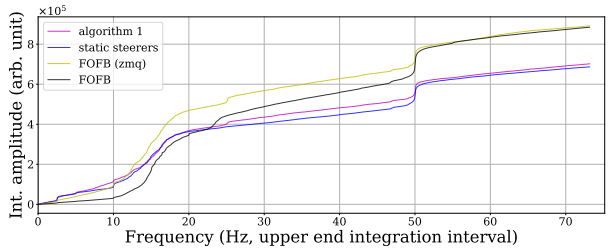


Figure 19: Integrated mean horizontal beam motion spectra comparison (test with surrogate model, 21/6/21, BESSY Archiver data).

$$J^c := \begin{pmatrix} \frac{\partial F^1}{\partial c^1_{t+1}} & \cdots & \frac{\partial F^m}{\partial c^1_{t+1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial F^1}{\partial c^l_{t+1}} & \cdots & \frac{\partial F^m}{\partial c^l_{t+1}} \end{pmatrix}$$

An averaged evaluation of matrices J^c with test data (approximating $\mathbb{E}_{(\mathbf{x}_{t-n},\ldots,\mathbf{x}_t,\mathbf{c}_{t+1}) \sim \text{test data}}[J^c\,|_{(\mathbf{x}_{t-n},\ldots,\mathbf{x}_t,\mathbf{c}_{t+1})}]$) can be visualised in Fig. 20. Notice that, as expected, the matrix structure is almost identical to the measured response matrix (Fig. 21).

Furthermore, a visualisation of the standard deviation of the evaluated matrices J^c (Fig. 22) gives us additional insights about which steerer-BPM interactions are captured by the surrogate model F in a more *non-linear* way: those entries presenting higher variance in the set of evaluated J^c's.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

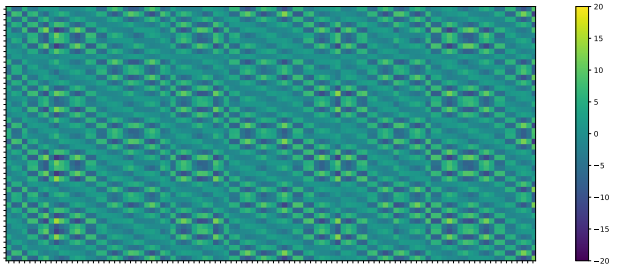


Figure 20: Average of matrices J^c evaluated at 4000 test points.

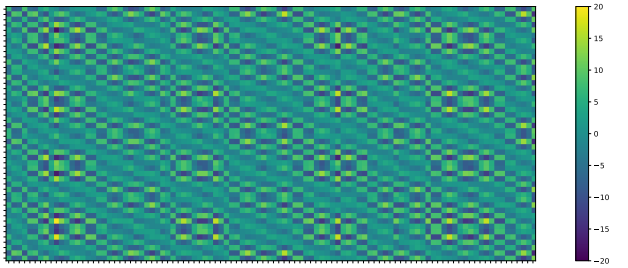


Figure 21: Horizontal response matrix measured at BESSY II (February 2021).

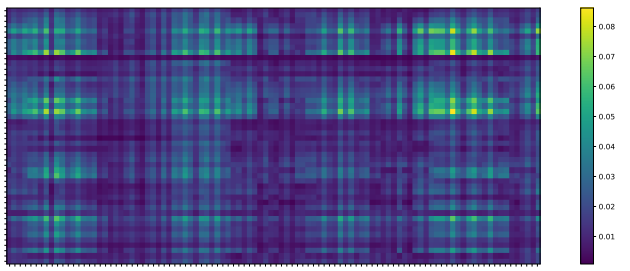


Figure 22: Standard deviation of matrices J^c evaluated at 4000 test points.

ANOMALY DETECTION WITH FEATURE ASSIGNATION

The use of anomaly detection algorithms for accelerator parameters is another field in current development at BESSY II. In particular we have started experimenting with Isolation Forests [10], whose simple and elegant foundations (outliers as points with *short path lengths*) have allowed us to develop additional anomaly scores that can be individually assigned to the input features. Our feature anomaly scores are conceived online and complement the information given by the global anomaly score: They should evaluate *how anomalous* the features of the current observation are. The idea is based on the philosophy of feature importance for fully randomized trees in [11] and similar to LFI in [12]. The results are comparable to SHAP [13] but with much better time performance for large data sets.

Given a point x, its node path in i-th tree T_i will be denoted as $\mathscr{P}_i(x) = \{N_0, N_1, \dots, N_l\}$. Notice that N_0 will be the root node and N_l a leaf. For each node N in T_i let $s(N)$ denote the number of samples assigned to this node after training. We can hence define a *splitting path* of a given point x in T_i that tracks the reduction of training samples along the path

$$\mathscr{S}_i(x) := \left\{ \left(1 - \frac{s(N_{j+1})}{s(N_j)} \right)^p \right\}_{N_j, N_{j+1} \in \mathscr{P}_i(x), j=1,\dots,l}$$

with $p > 0$ *sparsity* coefficient. If we denote as $F(N_j)$ the feature used for splitting at node N_j with $j = 0, \dots, l-1$, we can assign to it the j-th quantity within $\mathscr{S}_i(x)$. Therefore, given a point x and a feature F, we can define their *anomaly score* at tree T_i as

$$a_i(F, x) := \sum_j \{ s_j \in \mathscr{S}_i(x) \,|\, F(N_j) = F \}$$

and extend it naturally to all trees in the forest:

$$a(F, x) := \frac{1}{k} \sum_{i=1}^{k} a_i(F, x)$$

Figure 23 shows an anomaly detection application prototype including these feature anomaly scores and conceived as a first proof-of-concept. It corresponds to an isolation forest trained with ca. 125k data points along three weeks from 22 BESSY's top-up variables. Once the model has been trained, the application starts evaluating data points read in real-time and detects a decay of the global anomaly score, mainly assigned to a single variable (booster current per bunch). This value was indeed anomalous according to BESSY's predefined alert intervals; The algorithm had been able to recognize it without any previous information about these intervals.

Figure 23: Screenshot of the live application for anomaly detection at BESSY II: An anomaly score decay (lower right plot) is assigned in real time to the booster current per bunch (first bar in the upper plot).

CONCLUSION

The central use case presented in this paper (mitigation of harmonic orbit perturbations) was firstly faced with a model-free deep RL agent working with 6 BPMs and horizontal steerers that, as expected, improved stability in frequency domain but does not overcome traditional global methods in time domain. First steps towards model-based global optimisation with all 100 BPMs and 48 horizontal steerers keeping

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the time domain correction quality and improving also stability in frequency domain were introduced as well. These first model-based approaches are based on very accurate surrogate models encapsulating the inherent perturbations of the orbit. Finally, we have introduced further original ideas and a proof-of-concept for anomaly detection with feature assignation, whose first prototypes are ready for integration at the machine.

APPENDIX

- **Frameworks**: [14–16]
- **DDPG** - Hyperparameters:
  - **Actor**: feedforward network with three hidden layers (50-20-10 neurons), `ReLU` as activation function (output with `tanh`)
  - **Critic**: feedforward network with four hidden layers (50-(50+action)-20-10) neurons, `ReLU` as activation function, `Adam` as optimizer
  - **Learning**: $\gamma = 0.99$, target model update rate = 0.01, batch size = 32
  - **Exploration**: Ornstein-Uhlenbeck process ($\sigma = 0.05$, $\theta = 0.1$, no annealing), memory buffer with 20000 steps, short episodes (50 steps) followed by steerers random restart.
- **Surrogate model correction** - Hyperparameters:
  - **Model**: architecture plotted in Fig. 16, ReLU as activation function, linear output, `Adam` as optimizer
  - **Learning**: 16000 points for training, 4000 for test (gathered at 150 Hz), validation split 0.05, batch size = 32, 50 epochs.
  - **Steerer randomisation**: steerer intensities sampled from $\mathcal{N}(0, 5)$ (in mA) and actually set with probability $p = 0.005$.

REFERENCES

[1] P. Schnizer *et al.*, "Online Model Developments for BESSY II and MLS," in *Proc. IPAC'21*, Campinas, SP, Brazil, 2021, paper WEPAB317, pp. 3413–3416, doi:10.18429/JACoW-IPAC2021-WEPAB317

[2] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's Ahead: A Multi-Facility Collaboration for an a la Carte Software Project for Data Acquisition and Management," *Synchrotron Radiation News*, vol. 32, no. 3, pp. 19–22, 2019, doi:10.1080/08940886.2019.1608121

[3] W. Smith, S. Kazarski, R. Müller, P. Schnizer, S. Vadilonga, and L. V. Ramirez, "Status of Bluesky Deployment at BESSY II," presented at ICALEPCS'21 in Shanghai, China, paper FRBR03, this conference.

[4] L. V. Ramirez, G. Hartmann, T. Mertens, R. Müller, and J. Viefhaus, "Adding Machine Learning to the Analysis and Optimization Toolsets at the Light Source BESSY II," in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 754–760, doi:10.18429/JACoW-ICALEPCS2019-TUCPL01

[5] R. Mueller, R. Goergen, R. Lange, I. Mueller, and J. Rahn, "Introducing Fast Orbit Feedback at BESSY," in *Proc. ICALEPCS'09*, Kobe, Japan, 2009, paper THP059, pp. 773–775, https://jacow.org/icalepcs2009/papers/thp059.pdf

[6] O. Churlaud, "Localization and correction of orbit perturbations in BESSY II storage ring," M.S. thesis, TU Berlin, HZB, 2016.

[7] I. Agapov, G. Geloni, S. Tomin, and I. Zagorodnov, "OCELOT: A software framework for synchrotron light source and FEL studies," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 768, pp. 151–156, 2014, (c) Elsevier B.V., doi:10.1016/j.nima.2014.09.057

[8] J. Feikes, K. Holldack, P. Kuske, and R. Mueller, "Orbit Stability at BESSY," in *Proceedings of the 2005 Particle Accelerator Conference (PAC'05)*, 2005, pp. 2366–2368, doi:10.1109/PAC.2005.1591112

[9] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016, http://arxiv.org/abs/1509.02971

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422, doi:10.1109/ICDM.2008.17

[11] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, 2013, pp. 431–439, http://dl.acm.org/citation.cfm?id=2999611.2999660

[12] M. Carletti, M. Terzi, and G. A. Susto, *Interpretable Anomaly Detection with DIFFI: Depth-based Feature Importance for the Isolation Forest*, 2020.

[13] S. M. Lundberg *et al.*, "Explainable AI for Trees: From Local Explanations to Global Understanding," *CoRR*, vol. abs/1905.04610, 2019, http://arxiv.org/abs/1905.04610

[14] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015, https://www.tensorflow.org/

[15] F. Chollet, *Keras*, https://github.com/fchollet/keras, 2015.

[16] M. Plappert, *Keras-rl*, https://github.com/keras-rl/keras-rl, 2016.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

BAYESIAN TECHNIQUES FOR ACCELERATOR CHARACTERIZATION AND CONTROL

R. Roussel*, A. Edelen, C. Mayes, SLAC National Accelerator Laboratory, 94025 Menlo Park, USA
J. P. Gonzalez-Aguilera, Y.K. Kim, University of Chicago, 60637 Chicago, USA

Abstract

Accelerators and other large experimental facilities are complex, noisy systems that are difficult to characterize and control efficiently. Bayesian statistical modeling techniques are well suited to this task, as they minimize the number of experimental measurements needed to create robust models, by incorporating prior, but not necessarily exact, information about the target system. Furthermore, these models inherently take into account noisy and/or uncertain measurements and can react to time-varying systems. Here we will describe several advanced methods for using these models in accelerator characterization and optimization. First, we describe a method for rapid, turn-key exploration of input parameter spaces using little-to-no prior information about the target system. Second, we highlight the use of Multi-Objective Bayesian optimization towards efficiently characterizing the experimental Pareto front of a system. Throughout, we describe how unknown constraints and parameter modification costs are incorporated into these algorithms.

INTRODUCTION

Tuning accelerators to meet operational goals of a given facility is a time-consuming task that limits valuable beam time resources for experimental users. This process represents a difficult to solve optimization problem, where there are many free parameters and limited, expensive to conduct measurements available to diagnose target objectives. Accelerator optimization problems also exist in tightly constrained parameter spaces where large regions of parameter space prevent even simple measurements of the beam. Finally, due to the complexity of accelerator systems, measurements are often noisy and/or have large uncertainties.

Model based optimization methods have been shown to speed up convergence of optimizing black box problems, where derivative information about the target function is not accessible, making routine operations faster and previously impossible to solve problems solvable in realistic settings. Of particular interest is the use of Bayesian optimization (BO) techniques for solving optimization problems [1, 2] including experimental optimization of accelerators [3, 4]. Bayesian statistical models aim to represent measurements as probability distributions, instead of scalar values. This naturally lends itself to characterizing experimental accelerator measurements, which have inherent noise and uncertainty. Bayesian optimization explicitly takes these uncertainties into account when performing optimization, resulting in an algorithm that is robust to noise (an issue faced by many other types of algorithms) [5]. This method is especially proficient at *efficient* global optimization, since the Bayesian surrogate model encodes high level information about the target function behavior (such as function smoothness), allowing it to make accurate predictions about the function with limited data sets, thus significantly improving optimization performance over other methods.

Bayesian optimization consists of two elements, a statistical surrogate model that makes predictions about a given target function and an acquisition function which uses those predictions to choose future points in input space to measure. The surrogate model is usually chosen to be a Gaussian process (GP) [6]. This model treats the value of the target function at each point in input space $\mathbf{x}$ as a random variable taken from a normal distribution $f \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})^2)$ where $\mu(\mathbf{x})$ is the mean and $\sigma(\mathbf{x})$ is the standard deviation. Gaussian processes treat the joint probability distribution of the function values at each point in input space as a Multivariate normal distribution, specified by a mean function μ and a covariance matrix $\mathbf{\Sigma}$. We encode the expected behavior of the target function by specifying $\mathbf{\Sigma}$ via a kernel function $K(\mathbf{x}, \mathbf{x}')$ that describes how strongly function values at locations $\mathbf{x}$, $\mathbf{x}'$ are correlated with one another. A common class of kernel functions, known as "stationary kernels", are based solely on the distance between the two points, $||\mathbf{x}-\mathbf{x}'||$. We can then specify the expected smoothness of our function with a length-scale hyperparameter in specific kernels, such as the radial basis function (RBF) or Matern kernels [6]. Once experimental data is incorporated into the model, it can then be used to predict the function mean and corresponding uncertainty everywhere in the input domain.

Once a model is generated, we can then specify an acquisition function $\alpha(\mathbf{x})$ characterizes how valuable future observations are as a function of input parameters. For example, if we have high confidence in our model, we can choose to make measurements where the predicted function mean is at an extremum, thus heavily weighting "exploitation". On the other hand, if we wish to improve our understanding of the target function, we can place a high value on making observations in regions of high uncertainty, thus reducing the overall uncertainty of the model by heavily weighting "exploration". The most popular acquisition functions for single objective global optimization balances these two aspects, either implicitly using expected improvement over the best previously observed function value [7], or explicitly using an optimization hyperparameter [8].

In this work, we describe two acquisition functions that are specifically tailored to solve accelerator control problems. The first example describes "Bayesian exploration", an algorithm that enables automatic, efficient characterization of target functions, replacing the need for grid-like parameter

* rroussel@slac.stanford.edu

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

scans. The second is Multi-Objective Bayesian optimization (MOBO) which can be used to determine the entire Pareto front of a multi-objective optimization problem, as is often the case for control systems of large experimental facilities, using serialized measurements. We describe experimental demonstrations of each of these algorithms at two experimental facilities, the Linac Coherent Light Source (LCLS) and the Argonne Wakefield Accelerator (AWA). While our demonstrations are focused on the use of these algorithms in accelerator control systems, they can be easily used in a variety of other large experimental physics control systems.

BAYESIAN EXPLORATION

Due to the complex and time-consuming nature of accelerator diagnostics, characterization of beam response to input parameters is often limited to simple, uniformly spaced, grid-like parameter scans in one or two dimensions. This limitation results from the poor scaling of grid-like scans to higher dimensional spaces, where the number of samples needed grows exponentially with the number of input parameters. Furthermore, when attempting to characterize unknown systems it is often difficult to determine the ideal parameters of a grid scan that would result in an efficient and valid parameter scan.

The existence of tight constraints on which measurements are viable further complicates this process. For example, transverse beam size measurements on diagnostic screens are limited by the screen size, which in turn, imposes limits on the strength of upstream focusing magnet parameters. Simulation studies or extra measurements are needed beforehand to determine these limits. While these limits can be easily determined for a single parameter experimentally, it becomes infeasible to efficiently determine limits in higher dimensional input spaces, as they are often correlated with multiple parameters. Limitations such as these are shared among many types of control and optimization problems [9].

Finally, it is desirable to prevent rapid changes in accelerator input parameters during characterization. In some cases, it is temporally expensive to make changes in parameters, such as when mechanical actuators are used to change the phase of accelerating cavities. In other cases, fast feedback algorithms used in accelerator subsystems rely on adiabatic changes in external parameters to maintain system stability. Large jumps in parameter space can delay convergence of these feedback systems to stability or worse, cause them to fail entirely [10]. Practical experimental considerations such as these must be considered when automated sampling algorithms are used, which should strike a balance between costs associated with changing input parameters and the information gained.

To solve this problem, we developed an acquisition function to enable automated characterization of a target function in an efficient manner [11]. The acquisition function is given by

$$\alpha(\mathbf{x}, \mathbf{x}_0) = \sigma(\mathbf{x})\Psi(\mathbf{x}, \mathbf{x}_0) \prod_{i=1}^{N} P_i[g_i(\mathbf{x}) \geq h_i] \quad (1)$$

$$\Psi(\mathbf{x}, \mathbf{x}_0) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{x}_0)\right) \quad (2)$$

where $\sigma(\mathbf{x})$ is the standard deviation of the GP model, $\mathbf{x_0}$ is the most recently observed point, $\boldsymbol{\Sigma}$ is a diagonal, positive semi-definite matrix chosen by the experimenter and $P_i[g_i(\mathbf{x}) \geq h_i]$ represents the probability that the i'th operational constraint $g_i(\mathbf{x}) \geq h_i$ is satisfied. The terms of this acquisition function are as follows.

The first term $\sigma(\mathbf{x})$ is used to highly value points where model uncertainty is largest and has been shown to maximize information gain about the model [8]. When used with a GP kernel that has separate length scales for each free parameter in a process known as automatic relevance determination [6], this term will cause the optimizer to increase sampling frequency along the dimension with the shortest length scale, as seen in Fig. 1.

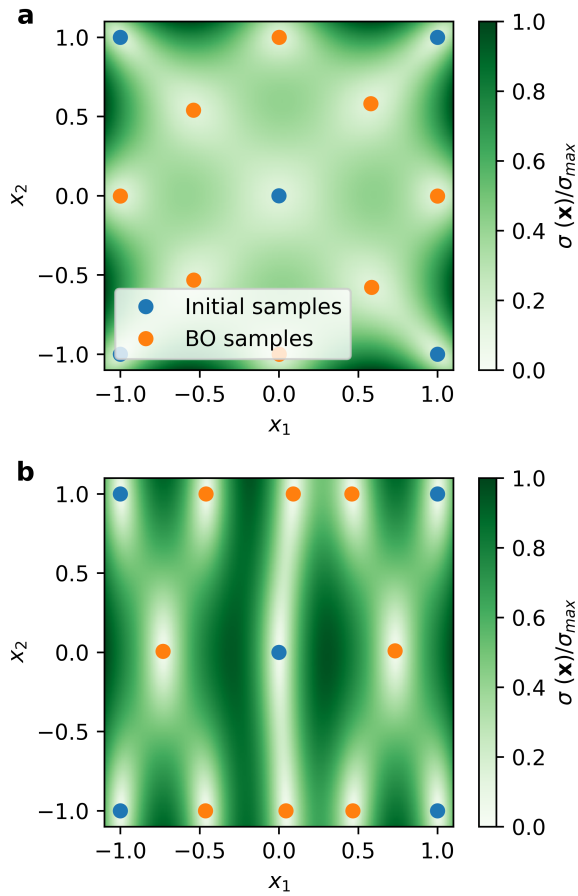


Figure 1: Plots showing Bayesian optimization sampling patterns depending on the kernel length scale using $\alpha(\mathbf{x}) = \sigma(\mathbf{x})$. Blue points are initial samples and orange points are determined via Bayesian optimization. (a) Length scales for both $[x_1, x_2]$ are set to 1. (b) Length scales for variables $[x_1, x_2]$ are set to $[0.25, 1]$. Reproduced from [11].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

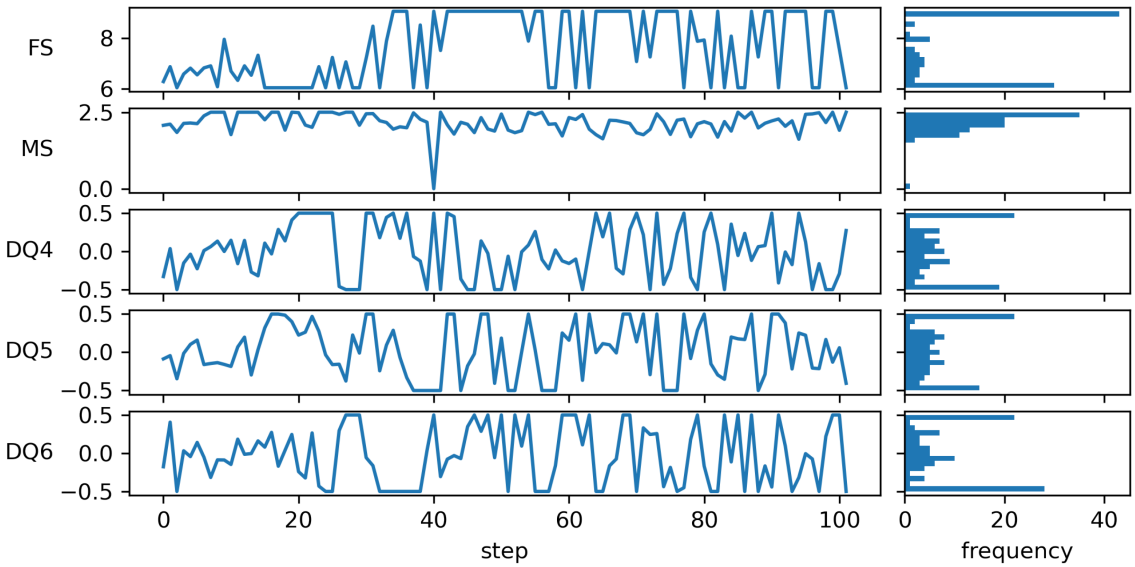


Figure 2: Parameter values (in arbitrary units) as a function of sample index during Bayesian exploration along with sample density histograms.

The second term $\Psi(\mathbf{x}, \mathbf{x_0})$ is a proximal weighting term, which biases the optimizer towards taking small steps in parameter steps. Another viable way to achieve this effect would be to place a hard limit on the maximum step size that the optimizer can take by setting the acquisition function to zero outside a given radius away from the last observed point. However, this negatively impacts the "exploration" advantages afforded by Bayesian optimization. By specifying a weighting that never reaches zero, we allow the optimizer to travel large distances in input space if it predicts a high potential value for the measurement.

Finally, the last term represents weighting due to constraining functions. For each constraint, we create a separate GP model that is used to calculate the probability that a given constraining function $g_i(\mathbf{x})$ satisfies $g_i(\mathbf{x}) \geq h_i$ where h_i is a constant. By weighting the acquisition function by these probabilities, it is less likely to propose points that violate any of the constraining functions, without prior knowledge of what those functions look like.

We conducted an experiment at the Argonne Wakefield Accelerator [12] to characterize the horizontal beam size at a diagnostic screen as a function of drive beam line parameters. These parameters included a solenoid around the photoinjector (FS), a solenoid just downstream of the photoinjector (MS) and three quadrupoles near the diagnostic screen (DQ4, DQ5, DQ6). The measurement was constrained by an upper bound on transverse beam size in both directions on the diagnostic screen and an upper bound on beam centroid distance from the screen center. We started the optimizer at a point in input space that satisfied the constraints and allowed it to explore for 100 iterations, taking 5 samples at each step and averaging the beam properties. At a repetition rate of 1 Hz, 100 iterations of the algorithm took less than 20 minutes, averaging less than 12 s per sample.

A trace of the sampled points in input space during exploration is seen in Fig. 2. Many of the parameters (FS, DQ4-6) did not have a strong effect on the bunch size over the given domain, which is inferred from the length scales of the fitted GP model for each parameter. As a result, most of the samples are on the domain boundaries and midpoints of each parameters' domain. Also note that these results imply that the next exploration run should happen over a larger domain if technologically feasible. On the other hand, we determined from its short length scale that the beam size was strongly correlated with the strength of the second solenoid (MS) relative to its input domain. Furthermore, we observe that only a narrow set of values for this solenoid satisfy the imposed constraints on the measurement. If we had attempted to characterize this input domain with a grid-like parameter scan, it is likely that a substantial number of measurements would have violated the constraints. Instead, our measurements using Bayesian exploration were valid about 80 percent of the time and sampling density was much higher for parameters that had strong correlations with our target measurement.

MULTI-OBJECTIVE BAYESIAN OPTIMIZATION

Many problems in accelerator physics optimization are multi-objective, where we must determine points in input space that optimally balance the trade-off between multiple, competing objectives, also known as the Pareto front. A popular set of techniques used to accomplish this are known as evolutionary algorithms. These algorithms, such as Non-dominated Sorting Genetic Algorithm II (NSGA-II) [13] or Multi-Objective Particle Swarm Optimization [14–16], are based on the generation of a diverse collection of candidate solutions, which are then observed via simulation or experiment, usually in a parallelized manner. The results from each observation are then sorted into non-dominated and dominated subsets. The non-dominated subset of can-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

didate solutions is used to produce the next "generation" of candidate solutions using a stochastic heuristic, which are then re-evaluated. The process is repeated over several generations until the non-dominated set of observations converges to a stationary Pareto front or the hypervolume has converged to a maximum value. It has been shown that these methods are well suited for solving accelerator design optimization problems [17, 18].

However, these algorithms are poorly suited for online accelerator optimization. Evolutionary type algorithms often rely on parallelized evaluation of the objective functions. However, they become practically inefficient when restricted to serialized evaluations, as is the case for online accelerator optimization. This inefficiency rises from evolutionary algorithms use of the binary classification metric of Pareto dominance to generate the next generation of potential observation candidates. As a result, this metric does not guarantee optimal expansion of the Pareto front, as it does not consider the relative hypervolume improvement of individuals in the non-dominated subset of candidates.

The Multi-Objective Bayesian Optimization (MOBO) algorithm [19] on the other hand, maximizes optimization efficiency by using an explicit calculation of the hypervolume improvement as an acquisition function to select the best candidates to expand the Pareto frontier. The hypervolume improvement $\mathcal{H}_I$ is defined as the increase in Pareto front hypervolume by adding a new observation $\mathbf{y}$ (see Fig. 3). Each objective is modeled using a GP surrogate model which can be used to predict the hypervolume improvement as a function of input parameters. By maximizing the hypervolume improvement acquisition function, MOBO can determine a single point that maximally increases the Pareto front hypervolume at every step when evaluated in a serialized manner, making it ideal for online accelerator optimization.

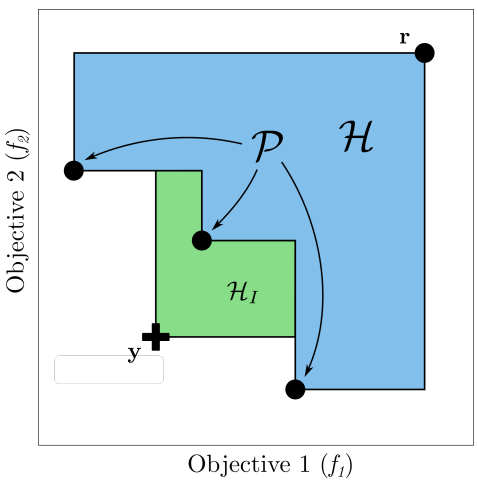


Figure 3: Cartoon of multi-objective optimization where each objective is to be minimized. Multi-objective optimization attempts to find a set of points known as the Pareto front $\mathcal{P}$ that dominate over a reference point $\mathbf{r}$ and any other observed points in objective space. The Pareto front hypervolume $\mathcal{H}$ (shown in blue) is the axis-aligned volume enclosed by the Pareto front and a reference point $\mathbf{r}$. Making a new observation $\mathbf{y}$, that dominates over points in the current Pareto front, leads to an increase in hypervolume (shown in green), referred to as the hypervolume improvement $\mathcal{H}_I$. Reproduced from [20].

The most common multi-objective acquisition function, expected hypervolume improvement (EHVI), is analogous to the popular single-objective expected improvement acquisition function [21]. This acquisition function calculates the average increase in hypervolume using the probability distribution of each objective function from the surrogate model. The EHVI acquisition function is formally defined as

$$\alpha_{EHVI}(\mu, \sigma, \mathcal{P}, \mathbf{r}) := \int_{\mathbb{R}^P} \mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{r}) \cdot \xi_{\mu,\sigma}(\mathbf{y}) d\mathbf{y} \quad (3)$$

where $\mathcal{P}$ is the current set of Pareto optimal points, $\mathbf{r}$ is the reference point, $\mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{r})$ is the hypervolume improvement from an observed point $\mathbf{y}$ in objective space, and $\xi_{\mu,\sigma}$ is the multivariate Gaussian probability distribution function with the GP predicted mean μ and standard deviation σ for each objective. A downside of this acquisition function is the calculation cost of determining the expected hypervolume improvement in high-dimensional objective spaces. A similar acquisition function, upper confidence bound hypervolume improvement (UCB-HVI) can be used as a cheaper to evaluate substitute with similar performance to EHVI [22].

We demonstrated the use of MOBO on optimizing the LCLS photoinjector [23]. Our goal was to determine the ideal trade-off between minimizing both the longitudinal bunch size σ_z and the vertical transverse beam size σ_y with respect to three tunable parameters, the gun solenoid, a focusing quadrupole magnet and a skew quadrupole magnet. The bounds of each free parameter were determined prior to experimentation from operator experience. The longitudinal bunch length was measured by a transverse deflecting cavity in the horizontal direction. Beam size measurements were conducted using optical transition radiation and statistics for each measurement were calculated from 5 samples each.

We began by randomly sampling 20 points in input space, producing the blue measurements in Fig. 4. We then used MOBO, with the UCB-HVI acquisition function to perform optimization for 20 iterations, initialized with data from the random search, shown in red in Fig. 4. Within these iterations MOBO was able to determine an approximate Pareto front (shown in the inset) that drastically improves over points found over random sampling. While this demonstration was a simple one, previous simulation results of optimizing the Argonne Wakefield Accelerator photoinjector with 7 objectives and 6 free input parameters show significant improvement over evolutionary algorithms commonly used by the accelerator community [20]. Furthermore, we also demonstrated that the same treatment of constraints and proximal biasing as was used in Bayesian exploration could also be incorporated into the MOBO acquisition function.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

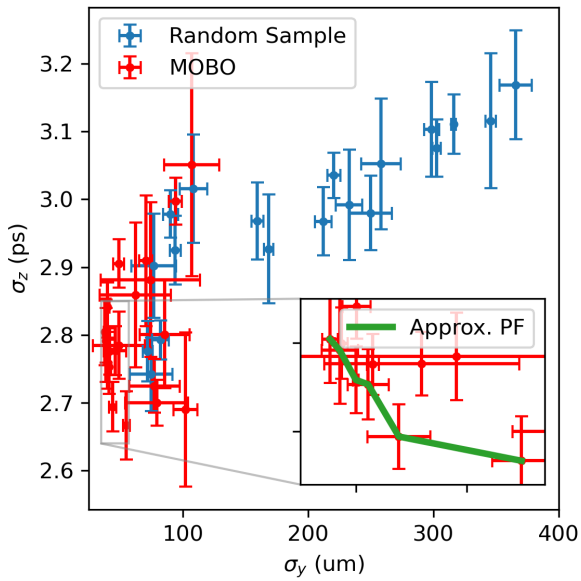


Figure 4: Results from running random search and then multi-objective Bayesian optimization to optimize the trade-off between electron bunch length σ_z and transverse beam size σ_y for the LCLS photoinjector. Free parameters included the gun solenoid, a focusing quadrupole and a skew quadrupole. Error bars denote 1σ error for each measurement due to jitter. Inset: Approximate Pareto front (PF) after 20 MOBO iterations.

Xopt: EASILY ACCESSIBLE IMPLEMENTATION OF ADVANCED ALGORITHMS

Using these algorithms on novel problems can take a significant amount of set-up and training to be successful. To reduce the upfront costs of using these and other complex algorithms for experimenters we have developed a flexible framework *Xopt* [24] for connecting advanced optimization algorithms to experimental control systems or simulations. Users need only to specify an algorithm by name and a python function that handles evaluation of target functions. Customization of optimization algorithms is also easily done, although their default configurations have been set to handle most problems off the shelf. *Xopt* has been used to both control the AWA accelerator and dispatch parallelized simulations on the Cori high performance computing cluster at NERSC [25]. The code is freely available at `https://github.com/ChristopherMayes/Xopt`.

CONCLUSION

In this work we have shown that Bayesian techniques can be used to efficiently characterize and optimize accelerators. They can navigate tightly constrained systems without prior knowledge regarding those systems. Furthermore, they can be modified by proximal biasing factors which promote adiabatic changes in controllable parameters, a requirement for efficient and stable optimization of large experimental systems.

ACKNOWLEDGEMENTS

We would like to thank the operations staff at both LCLS and AWA for their contributions to this work. This work was supported by the U.S. Department of Energy, under DOE Contracts No. DE-AC02-76SF00515, DE-AC02-06CH11357, the Office of Science, Office of Basic Energy Sciences as well as the Office of High Energy Physics, and U.S. National Science Foundation under Award No. PHY-1549132, the Center for Bright Beams.

REFERENCES

[1] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," in *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, Jan. 2016. doi:10.1109/JPROC.2015.2494218

[2] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, "Bayesian Optimization for Adaptive Experimental Design: A Review," in *IEEE Access*, vol. 8, pp. 13937-13948, 2020. doi:10.1109/ACCESS.2020.2966228

[3] J. Duris, D. Kennedy, A. Hanuka, J. Shtalenkova, A. Edelen, P. Baxevanis, A. Egger, T. Cope, M. McIntire, S. Ermon, and D. Ratner, "Bayesian Optimization of a Free-Electron Laser," *Phys. Rev. Lett.*, vol. 124, p. 124801, 2020. doi:10.1103/PhysRevLett.124.124801

[4] J. Kirschner, M. Mutný, N. Hiller, R. Ischebeck, and A. Krause, "Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces," in *Proc. Int. Conf. on Machine Learning*, vol. 97, pp. 3429-3438, 2019. https://arxiv.org/abs/1902.03229v2

[5] J. Snoek, H. Larochelle, and R.P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Ed. New York, NY, USA: Curran Associates, Inc., 2012, pp. 2951–2959.

[6] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*, Adaptive computation and machine learning. Cambridge, MA, USA: MIT Press 2006.

[7] J. Mockus, V. Tiesis, and A. Zilinskas, "The Application of Bayesian Methods for Seeking the Extremum," in *Towards Global Optimisation 2*, L.C.W.Dixon and G.P. Szego, Ed. North-Holland, pp. 117-129, 1978. https://www.researchgate.net/publication/248818761_The_application_of_Bayesian_methods_for_seeking_the_extremum

[8] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," in *Proceedings of the 27 th International Conference on Machine Learning (ICML'10)*, Haifa, Israel, 2010, pp. 1015–1022, Madison, WI, USA, June 2010. Omnipress. https://icml.cc/Conferences/2010/papers/422.pdf

[9] E.A. Baltz, E. Trask, M. Binderbauer, M. Dikovsky, H. Gota, R. Mendoza, J.C. Platt, and P.F. Riley, "Achievement of Sustained Net Plasma Heating in a Fusion Experiment with the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Optometrist Algorithm," *Scientific Reports*, vol. 7, p. 6425, July 2017. doi:10.1038/s41598-017-06645-7

[10] A.L. Edelen, S.G. Biedron, B.E. Chase, D. Edstrom, S.V. Milton, and P. Stabile, "Neural Networks for Modeling and Control of Particle Accelerators," *IEEE Transactions on Nuclear Science*, vol. 63, pp. 878–897, 2016. doi:10.1109/TNS.2016.2543203

[11] R. Roussel, J.P. Gonzalez-Aguilera, Y.-K. Kim, E. Wisniewski, W. Liu, P. Piot, J. Power, A. Hanuka, and A. Edelen, "Turn-key constrained parameter space exploration for particle accelerators using Bayesian active learning," *Nature Communications*, vol. 12, p. 5612, 2021. doi:10.1038/s41467-021-25757-3

[12] M.E. Conde, S.P. Antipov, D.S. Doran, W. Gai, Q. Gao, G. Ha, *et al.*, "Research Program and Recent Results at the Argonne Wakefield Accelerator Facility (AWA)," in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 2885–2887. doi:10.18429/JACoW-IPAC2017-WEPAB132

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002. doi:10.1109/4235.996017

[14] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proc. of the IEEE Int. Conf. on Neural Networks (ICNN'95)*, vol. 4, pp. 1942-1948, 1995. doi:10.1109/ICNN.1995.488968

[15] X.B. Huang and J. Safranek, "Nonlinear dynamics optimization with particle swarm and genetic algorithms for SPEAR3 emittance upgrade," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 757, pp. 48–53, September 2014. doi:10.1016/j.nima.2014.04.078

[16] X. Pang and L.J. Rybarcyk, "Multi-objective particle swarm and genetic algorithm for the optimization of the LANSCE linac operation," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 741, pp. 124–129, March 2014. doi:10.1016/j.nima.2013.12.042

[17] Y.J. Li, W.X. Cheng, L. H. Yu, and R. Rainer, "Genetic algorithm enhanced by machine learning in dynamic aperture optimization," *Phys. Rev. Accel. Beams*, vol. 21, p. 054601, May 2018. doi:10.1103/PhysRevAccelBeams.21.054601

[18] N. Neveu, L. Spentzouris, A. Adelmann, Y. Ineichen, A. Kolano, C. Metzger-Kraus, C. Bekas, A. Curioni, and P. Arbenz, "Parallel general purpose multiobjective optimization framework with application to electron beam dynamics," *Phys. Rev. Accel. Beams*, vol. 22, p. 054602, May 2019. doi:10.1103/PhysRevAccelBeams.22.054602

[19] M. Emmerich, K. Yang, A. Deutz, H. Wang, and C.M. Fonseca, "A Multicriteria Generalization of Bayesian Global Optimization," in *Advances in Stochastic and Deterministic Global Optimization*, P. Pardalos, A. Zhigljavsky, J. Žilinskas, Eds. Springer Optimization and Its Applications, vol. 107, pp. 229–242. Springer International Publishing, Cham. doi:10.1007/978-3-319-29975-4_12

[20] R. Roussel, A. Hanuka, and A. Edelen, "Multiobjective Bayesian optimization for online accelerator tuning," *Phys. Rev. Accel. Beams*, vol. 24, p. 062801, June 2021. doi:10.1103/PhysRevAccelBeams.24.062801

[21] M.T.M. Emmerich, A.H. Deutz, and J.W. Klinkenberg, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," *IEEE Congress of Evolutionary Computation (CEC)*, pp. 2147-2154, 2011. doi:10.1109/CEC.2011.5949880

[22] M.T.M. Emmerich, K. Yang, and A.H. Deutz, "Infill Criteria for Multiobjective Bayesian Optimization," in *High-Performance Simulation-Based Optimization. Studies in Computational Intelligence*, T. Bartz-Beielstein, B. Filipič, P. Korošec, E.-G. Talbi, Eds. Cham, Springer International Publishing, vol. 833, pp. 3–16, 2020. doi:10.1007/978-3-030-18764-4_1

[23] R. Akre, D. Dowell, P. Emma, J. Frisch, S. Gilevich, G. Hays, Ph. Hering, R. Iverson, C. Limborg-Deprey, H. Loos, A. Miahnahri, J. Schmerge, J. Turner, J. Welch, W. White, and J. Wu, "Commissioning the Linac Coherent Light Source injector," *Phys. Rev. ST Accel. Beams*, vol. 11, p. 030703, March 2008. doi:10.1103/PhysRevSTAB.11.030703

[24] C. Mayes, R. Roussel, and H. Slepicka, *ChristopherMayes/Xopt:* Xopt v0.5.0, October 2021. doi:10.5281/zenodo.5559141

[25] K. Antypas, N. Wright, N.P. Cardo, A. Andrews, and M. Cordery, "Cori: A Cray XC Pre-Exascale System for NERSC," in *Cray User Group Proceedings (CUG2014)*, Cray, 2014. https://cug.org/proceedings/cug2014_proceedings/includes/files/pap211.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

MACHINE LEARNING BASED MIDDLE-LAYER FOR AUTONOMOUS ACCELERATOR OPERATION AND CONTROL

S. Pioli*, B. Buonomo, F. Cardelli, P. Ciuffetti, L. G. Foggetta, C. Di Giulio,
D. Di Giovenale, G. Piermarini, INFN-LNF, Frascati, Italy
V. Martinelli, INFN-LNL, Legnaro, Italy

Abstract

The Singularity project, led by National Laboratories of Frascati of the National Institute for Nuclear Physics (INFN-LNF), aim to develop automated machine-independent middle-layer to control accelerator operation through machine learning (ML) algorithms like Reinforcement Learning (RL) and Clustering, integrated with accelerator sub-systems. In this work we will present the actual LINAC control system and the necessary effort to implement the architecture and the middle-layer necessary to develop autonomous operation control with RL algorithms together with the fault detection capability improved by Clustering approach as for waveguides or accelerator sections breakdown. Results of the first tentative operation of Singularity on the LINAC system will be reported.

INTRODUCTION

In this paper we will present our effort to integrate a Machine Learning (ML) based middle-layer integrated in the DAΦNE LINAC in order to demonstrate the feasibility of autonomous operation driven by RL algorithm together with Clustering fault detection methods to identify breakdown activities.

The main obstacle to implement the Singularity project [1] in the DAΦNE LINAC is related to the time available for the development and test the system in an operating accelerator for 4500 hours per year. The COVID periods with the reduced activities permits to implement the necessary step to provides at Singularity the data to implements the algorithms.

In the first part of this work introduction of the LINAC elements will be provided and an overview on the the actual control system is shown to introduce the architecture of the system and the middle layer implemented to provides the data to Singularity.

In the second part of the work description of RL algorithms and Singularity middle layer will presented and related integration and performances obtain on off-line operation will be presented and discussed.

THE DAΦNE LINAC

The DAΦNE injector is composed by a ~60 m long Linac that produces and accelerates up to the collider operation energy (510 MeV) both the positron and electron beams. It has been designed and built by the USA firm TITAN BETA and commissioned by the INFN-LNF staff [2]. In Fig. 1 shows

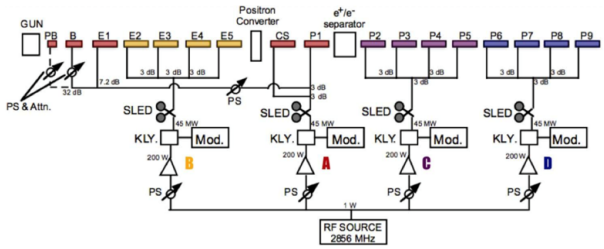


Figure 1: The LINAC layout.

the LINAC RF layout. The injector subsystem includes a thermionic electron gun, a prebuncher and a buncher. The sections performing at S-band working at 2856 MHz, and are powered by 4 klystrons Thomson TH2128C, with nominal output power of 45 MW, each one equipped with a SLED, the SLAC type pulse compressor device. The performances of the LINAC are summarized in Fig. 2.

	Design	Operational
Electron beam final energy	800 MeV	510 MeV
Positron beam final energy	550 MeV	510 MeV
RF frequency	2856 MHz	
Positron conversion energy	250 MeV	220 MeV
Beam pulse rep. rate	1 to 50 Hz	1 to 50 Hz
Beam macropulse length	10 nsec	1.4 to 300 nsec
Gun current	8 A	8 A
Beam spot on positron converter	1 mm	1 mm
norm. Emittance (mm. mrad)	1 (electron) 10 (positron)	< 1.5
rms Energy spread	0.5% (electron) 1.0% (positron)	0.5% (electron) 1.0% (positron)
electron current on positron converter	5 A	5.2 A
Max output electron current	>150 mA	500 mA
Max output positron current	36 mA	85 mA
Trasport efficiency from capture section to linac end	90%	90%
Accelerating structure	SLAC-type, CG, 2π/3	
RF source	4 x 45 MWp sledded klystrons TH2128C	

Figure 2: The LINAC beam performances.

After an upgrade on the gun [3] all the parameters as bunch duration and the other gun parameters and RF power in the RF guide distribution and in the sections could be controlled by klystron voltage set and the low level RF input of each klystron (power and phase) and the prebuncher, buncher, capture section power and phases could be set.

The focusing system varies its conformation according to the requirements of the portion of the LINAC interested.

* stefano.pioli@lnf.infn.it

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

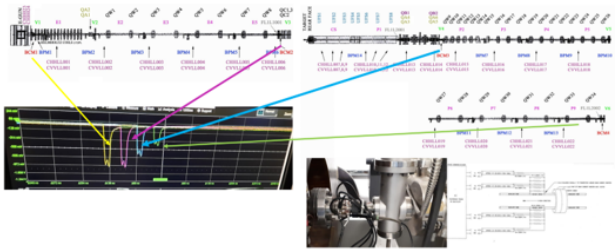


Figure 3: The LINAC magnet elements and diagnostics.

An easy way to describe this system is to follow a particle beam from the gun up to the LINAC end. More than 30 quadrupoles and a network of more than 20 vertical and horizontal correctors, coupled on each of the accelerating section approximately, permits the LINAC orbit correction. All the power supply of this magnets system are controlled by the LINAC control system and part by DAFNE control system.

The LINAC beam diagnostics system which includes a total of 14 beam position monitors BPM, 4 beam current monitors (BCM) and 4 beam profile monitors (flags). The BPMs are acquired by a multiplexed PXI independent system and the WCM by oscilloscope by a HP Multiplexer.

The position monitors are of the capacitive type with four electrodes, see Fig. 3. The voltages induced by the beam on these electrodes, properly combined, give the transverse position of the beam.

The current monitors are of the resistive wall type. The vacuum chamber continuity is interrupted by a ceramic gap and several resistors uniformly distributed around the gap create a resistive bridge, where the beam image current on the vacuum chamber can flow through. By monitoring the voltage drop on the resistors it is possible to derive the current value. Moreover the frequency response of the monitor is sufficient to reproduce the shape of the macrobunch (10 ns) with a good fidelity. Several examples of the current monitor signal are showed in Fig. 3.

To check the energy and the charge of the beam at the and of the LINAC a pulsed magnet drive one of the 50 bunch each second in the hodoscope spectrometer.

THE LINAC CONTROL SYSTEM

A distributed architecture has been chosen for the LINAC control system. These provide the feature to run all or part of the system individually for test and trouble-shooting, and being able to co-locate a subsystem with it's controls.

Originally the control system is operated through a control computer located remotely from the LINAC. An Apple Mac II run National Instruments LabVIEW originally and after a few upgrade in 2013 a server DELL with a virtual machine where the original LabVIEW 3.0 software was upgrade to LabVIEW 2010 on CentOS 7. The control software provides GUI shown in Fig. 4 and the operator full control of all functions and communicate with the CAMAC control system

Figure 4: The LINAC control system GUI.

via an IEEE-488 (GPIB) fiber-optical data link to isolate the computer from the control system.

A remote manual control panel is provided to perform limited hardwire commands and includes hardwire interlock functions. The control computer allow data logging of the system parameters with no impact on system performance. A CAMAC based interface was selected in 1996 for this system on the basis of performance, module availability, and familiarity with programming and operation of CAMAC based systems.

The CAMAC crate and associated controller, GPIB interfaces, and all digital and analog modules are located in the local control rack. The Command and Control buses handles all of the system wide commands and response from all of the subsystems. The Auxiliary buses is used for subsystem specific specific commands and response for the various operational configuration. All of the analog control and monitor signals are not bus-connected, but run direct to/from the required location in the shielded twisted pair cables, using various interface chassis as terminations points. Al the system under control are shown in Fig. 5.

The integration of all the systems and diagnostics are done following the schema represented in Fig. 6 follow the methodology presented in [4] to send and communicate with the Singularity AI. This system allows to integrate different data sources from the LINAC control system with the BPM system, WCM system and DAΦNE control system [5] and to send the command with the same interface.

THE REINFORCEMENT LEARNING APPROACH

According to Sutton's [6], Reinforcement Learning (RL) is a learning technique to map situations onto actions in order to maximize a numerical reward signal. The learner is not told which actions to perform, but must figure out through trial and error which actions yield the greatest reward by trying them. In the most interesting and challenging cases, the actions may affect not only the immediate reward, but also the next situation and through that, all subsequent rewards. These two characteristics (trial-and-error search and delayed

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

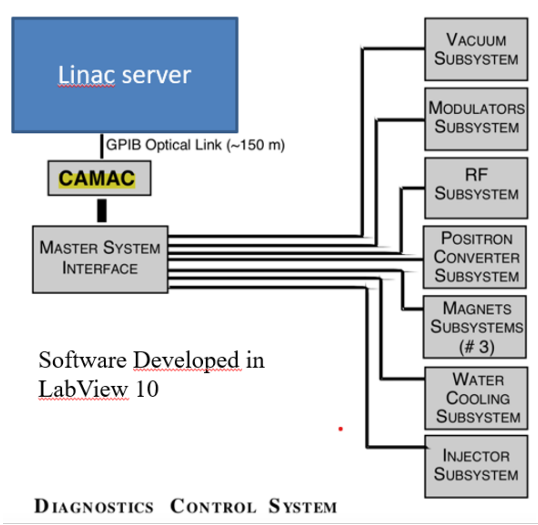


Figure 5: The LINAC control system.

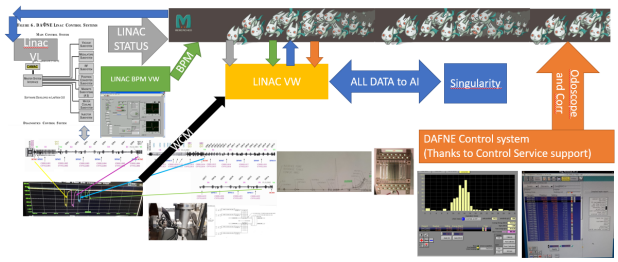


Figure 6: The LINAC control system and diagnostic acquisition.

reward) are the two most important distinguishing features of reinforcement learning.

Reinforcement learning is distinct from supervised learning, the type of learning studied in most current research in machine learning. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. Each example is a description of a situation along with a specification (the label) of the correct action that the system should take in that situation, often to identify a category to which the situation belongs. The goal of this type of learning is for the system to extrapolate or generalize its responses so that it acts correctly in situations not included in the training set.

One of the challenges that reinforcement learning faces is the trade-off between exploration and exploitation that does not exist in other types of learning. To obtain a high reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in obtaining a reward. However, in order to discover such actions, it must also try actions that it has not previously selected. The agent has to exploit what it has already experienced in order to obtain a reward, but it must also has to explore in order to make a better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions and gradually prefer those that seem best to it.

Q-learning [7] is a form of model-free reinforcement learning. It can also be considered as a method of asynchronous dynamic programming. It provides agents with the opportunity to learn to act optimally in Markovian domains, Fig. 7, by experiencing the consequences of their actions without having to create maps of the domains. Learning proceeds similarly to Sutton's [6] method of temporal differences (TD): an agent tries an action in a given state and evaluates its consequences based on the immediate reward or penalty it receives and its estimate of the value of the state it is placed in. By repeatedly trying all actions in all states, it learns which actions are best overall, judged by the long-run discounted reward according to the Bellman transition function.

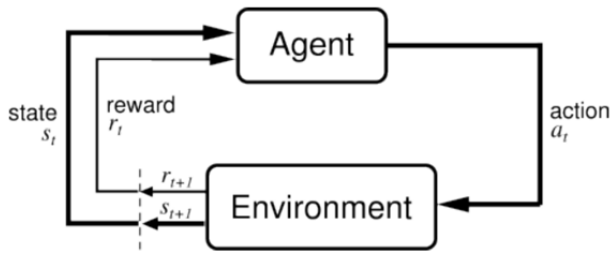


Figure 7: Markov Decision Process.

In this case, the learned action value function Q is a direct approximation to q*, the optimal action value function, regardless of the strategy followed. This drastically simplifies the analysis of the algorithm and allows for early convergence proofs. The policy still has an impact as it determines which state-action pairs are visited and updated. However, all that is required for proper convergence is that all pairs continue to be updated.

Under this assumption and a variant of the usual stochastic approximation conditions for the sequence of step-size parameters, Q converge to q* with probability 1. The Q-learning algorithm is presented below in procedural form in Fig. 8.

CONTROL SYSTEM INTEGRATION & FAULT DETECTION

Mem-cached live data from LINAC control system is acquired from Singularity middle-layer while a REST communication channel allow to change devices setpoint.

In order to allow the proper data validation for ML algorithm, interlock signals are acquired from the CAMAC in order to break and restore operation when a fault happens. Anyway not all the fault trigger an interlock like RF anomaly that should be processed through a data analysis to identify breakdown phenomena and stop the autonomous operation while the issue is occurring.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Figure 8: Q-learning pseudo-code iteration.

In order to handle this kind of RF events a Clustering algorithm have been involved to automatically recognize pattern inside the data so as to analyze the collected data without their labels. Using this advantage, a density based clustering fault diagnosis method have been used to deal with such RF breakdown issues, in which the labeled data are limited.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [8] algorithm requires two input parameters, a radius and minimum number used to define a density threshold in the data space. DBSCAN is an iterative algorithm which iterates over the objects in the dataset, analyzing their neighborhood. If there are more than certain number of objects whose distance from the considered object is less than the reference, then the object and its neighborhood originate a new cluster. DBSCAN is effective at finding clusters with arbitrary shape, and it is capable of identifying outliers as a low density area in the data space.

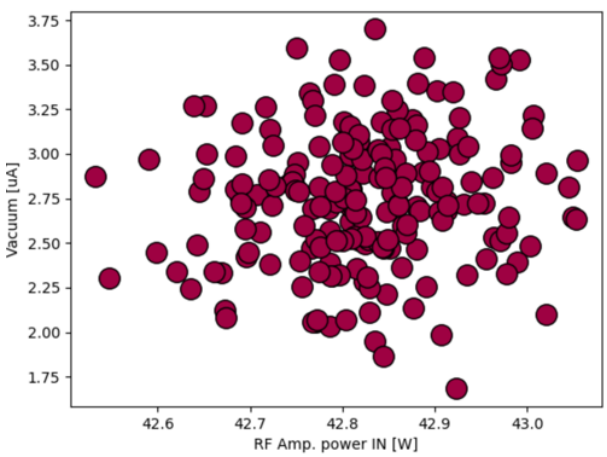


Figure 9: DBSCAN output in normal operation which identify one cluster.

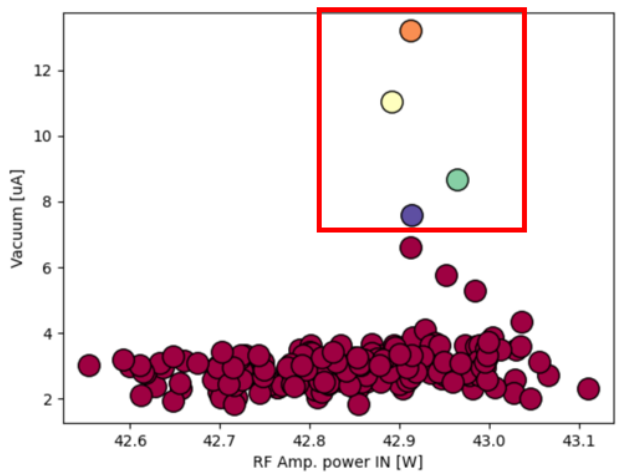


Figure 10: DBSCAN output in breakdown events which identify several clusters.

Thanks to this unsupervised learning technique have been possible control n-dimensional system identified by the evolution of RF power signal and all the vacuum ion pumps current trends in order to identify breakdown. The algorithm is setup to expect one cluster made of a reasonable distribution of point from RF power and ion pumps current distribution, as in Fig. 9. The minimum number of point to identify a cluster is set one in order to allow the classification of fault when just one ion pump show an anomalous behavior. When a vacuum readback moves out from the cluster, as in Fig. 10, this is identified as a second cluster and trigger the identification of a breakdown event stopping the execution of RL algorithm. This process is performed for all the RF sources of the LINAC and all the related vacuum ion pumps.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

AUTONOMOUS OPERATION TOOLS

Two tools for autonomous operation have been developed in order to handle main tasks, really time consuming, which involves the operators team to reach required performances for the DAΦNE complex and the BTF.

As general comment, all the tools have been developed to be capable of machine independence from the accelerator layout and to be invariant to performance drift of aged devices. The combination of these two conditions allow to use these tools independently on different new-one or legacy accelerator just configuring the schematic lattice and operating parameter of each device. The algorithm, during the training phase, will learn how to drive each device identifying for each setpoint the contribute to reach the goal state.

RF Sources Energy Tuning

According with definition of a Markovian Decision Process (MDP), the scope of this tool is the energy tuning of the LINAC beam through the control of two RF Sources (power and phase) after the positron converter and the monitoring the hodoscope at the end of the LINAC which provide the beam energy, as shown in Fig. 11.

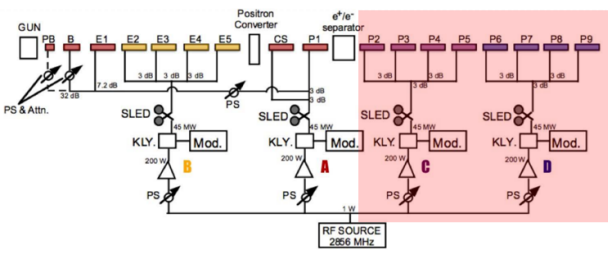


Figure 11: Highlighted in red the RF sources and related distribution controlled by the RF energy tuning tool.

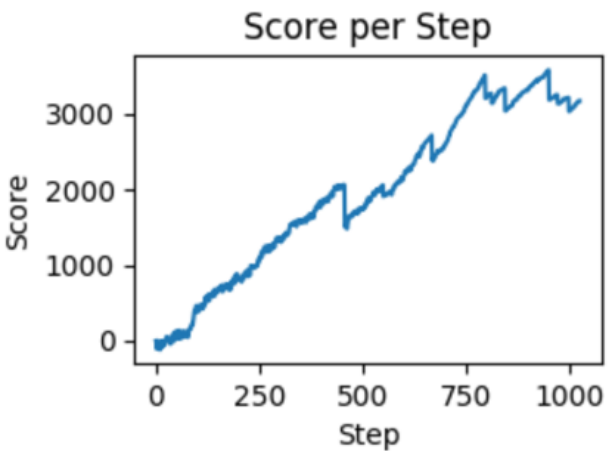


Figure 12: Score trend during each step of the 300 episodes training.

We identify for each variable of the environment operative ranges and the related accuracy. So the state-action matrix comes from the combination of all the possible action of the four variable parameters (power and phase of the two RF sources) with all the possible states (hodoscope energy readback). The Q-learning agent provide variable reward in function of the different between the current beam energy and the target beam energy.

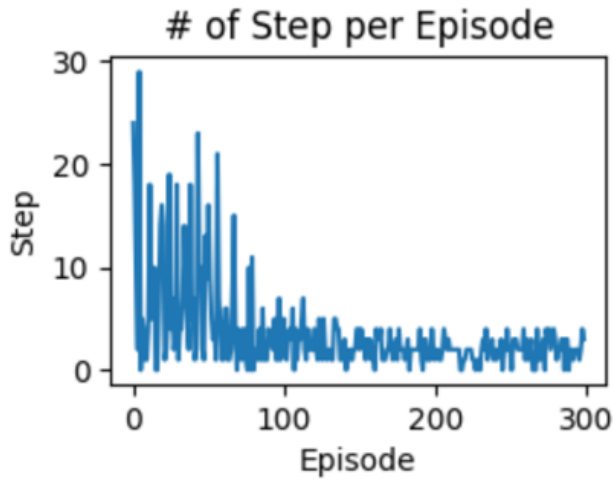


Figure 13: Number of Steps during each one of the 300 episodes training.

A simulated run with off-line data have been performed in order to test the algorithm over 300 episode. As shown in Figs. 12 and 13, the algorithm converge to Q* within 1000 steps equivalent to about one day of training on the LINAC (taking in consideration the required time to perform and stabilize a command with the CAMAC control system).

Beam Charge Optimization

The scope of this tool is optimize the charge of the LINAC beam through the control of 8 quadrupole magnets power supply, after the positron converter, and 2 beam current monitors (BCM) monitoring the beam current up to the end of the LINAC, as shown in Fig. 14.

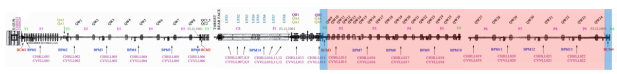


Figure 14: Highlighted in red the distribution of quadrupole magnets while in blue the position of the BCMs controlled by the Beam Current Optimization tool.

We identify for each variable of the environment operative ranges and the related accuracy. So the state-action matrix comes from the combination of all the possible action of the 8 variable parameters (the current setpoint of each power supply) with all the possible states (percentage of current transported through the two BCMs). The Q-learning agent provide variable reward in function of the different between the ratio of current transported and the target beam current ratio.

A simulated run with off-line data have been performed in order to test the algorithm over 300 episode. As shown in Figs. 15 and 16, the algorithm converge to Q* within 2000 steps equivalent to about two days of training on the LINAC

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

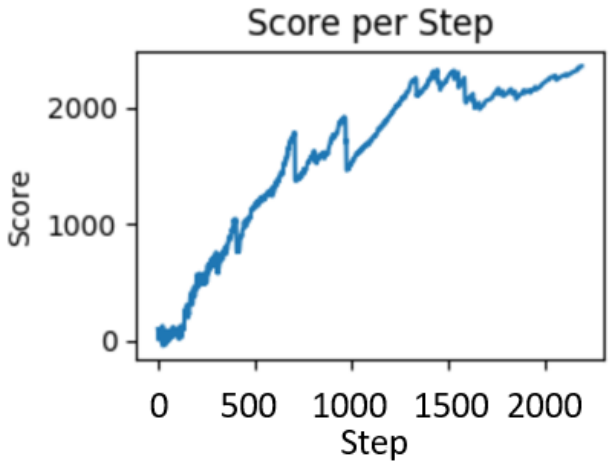


Figure 15: Score trend during each step of the 300 episodes training.

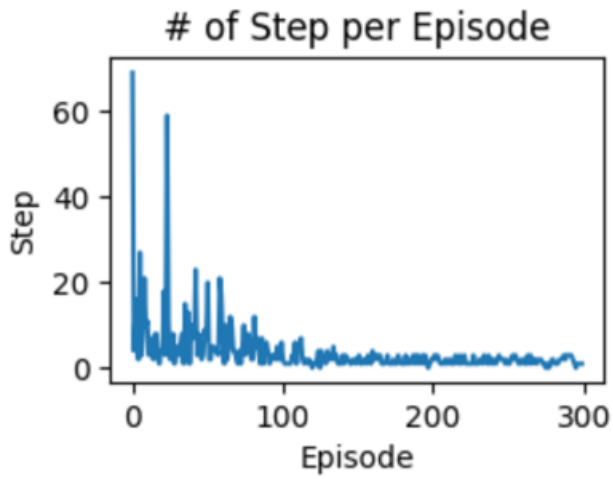


Figure 16: Number of Steps during each one of the 300 episodes training.

(taking in consideration the required time to perform and stabilize a command with the CAMAC control system).

CONCLUSION

The work presented in this paper show the progress in the development of Machine Learning based middle layer focused on Reinforcement Learning tool for automated operations.

Design of the DAΦNE LINAC control system, the first testing site, have been introduced together with the architecture of the Q-learning algorithm.

Validation process on off-line data highlight not only the feasibility of presented tools but also the capability to be trained on a LINAC with a small effort of few days of dedicated beam shift.

Next steps for the use of ML tools in operation will pass through a test shift to be scheduled in next months and an upgrade of the state-action matrix to a Deep Q-Network (DQN) to optimize memory usage.

REFERENCES

[1] S. Pioli, "Expression of Intent Singularity", INFN-19-05/LNF Technical Note

[2] F. Sannibale, M. Vescovi, R. Boni, F. Marcellini, and G. Vignola, "DAΦNE Linac Commissioning Results", DAFNE-NOTE-BM-02, http://www.lnf.infn.it/acceleratori/dafne/NOTEDAFNE/BM/BM-2.pdf

[3] B. Buonomo, L. G. Foggetta, and G. Piermarini, "New Gun Implementation and Performance of the DAΦNE LINAC", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, pp. 1546–1548, doi:10.18429/JACoW-IPAC2015-TUPWA056

[4] L.G. Foggetta, M. Belli, B. Buonomo, F. Cardelli, R. Ceccarelli, A. Cecchinelli, *et al.*, "The Extended Operative Range of the LNF LINAC and BTF Facilities", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 3987–3990. doi:10.18429/JACoW-IPAC2021-THPAB113

[5] G. Di Pirro, C. Milardi, A. Stecchi, and L. Trasatti, "DANTE: control system for DAΦNE based on Macintosh and LabVIEW", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 352, pp. 455-457, 1994. doi:10.1016/0168-9002(94)91568-7

[6] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction", 2018, MIT Press, Cambridge, MA, USA.

[7] C. J. C. H. Watkins and P. Dayan, "Q-learning", *Mach Learn*, vol. 8, pp. 279–292, 1992. doi:10.1007/BF00992698

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, pp. 226-231, 1996.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

MACHINE LEARNING BASED TUNING AND DIAGNOSTICS FOR THE ATR LINE AT BNL

J. P. Edelen*, K. Bruhwiler, E. Carlin, C. C. Hall, RadiaSoft LLC, Boulder, CO, USA
K. A. Brown, V. Schoefer, Brookhaven National Laboratory, Upton, NY, USA

Abstract

Over the past several years machine learning has increased in popularity for accelerator applications. We have been exploring the use of machine learning as a diagnostic and tuning tool for the transfer line from the AGS to RHIC at Brookhaven National Laboratory. In our work, inverse models are used to either provide feed-forward corrections for beam steering, or as a diagnostic to illuminate quadrupole magnets that have excitation errors. In this paper we present results on using machine learning for beam steering optimization for a range of different operating energies. We also demonstrate the use of inverse models for optical error diagnostics. Our results are from studies that use both simulation and measurement data.

INTRODUCTION

Machine learning (ML) has seen a significant growth in its adoption for widespread applications. In particle accelerators ML has been identified as having the potential for significant impact on modeling, operation, and controls [1,2]. These techniques are attractive due to their ability to model nonlinear behavior, interpolate on complicated surfaces, and adapt to system changes over time. This has led to a number of dedicated efforts to apply ML, and early efforts have shown promise.

For example, neural networks (NNs) have been used as surrogates for traditional accelerator diagnostics to generate non-interceptive predictions of beam parameters [3, 4]. Neural networks have been used for a range of machine tuning problems utilizing inverse models [5, 6]. When used in conjunction with optimization algorithms neural networks have demonstrated improved switching times between operational configurations [7]. Neural network surrogate models have also been demonstrated to significantly speed up multi-objective optimization of accelerators [8]. Additionally, ML has been of interest for anomaly detection, using autoencoders, for root cause analysis [9], and for outlier detection, using large data-sets of known good operational states [10].

In this work we seek to apply ML methods — for both tuning and anomaly detection — on the AGS to RHIC transfer line at Brookhaven National Laboratory. Specifically, we employ the use of inverse models for these applications. The application of inverse models for anomaly detection is a burgeoning area of research in many other fields that has not seen much attention in particle accelerators. Here we present our work towards implementing inverse models to detect errors in quadruples using only beam position monitors and corrector data. We will demonstrate the utility of this approach using a toy model, and then show how it scales to a larger system such as the AGS to RHIC transfer line. We will then show results of training inverse models using data from the machine and discuss future work for this effort.

* jedelen@radiasoft.net

THE ATR LINE

The transfer line between the Alternating Gradient Synchrotron (AGS) and RHIC, or the so-called the ATR line [11, 12], must be retuned for different energies when RHIC changes its operating point. The transfer line controls the orbit matching, optics matching, and dispersion matching of the beam into RHIC. The transfer line is broken down into four sections. The U and W lines are seen by all beams entering RHIC while the X and Y lines are used for injection into the Blue and Yellow rings respectively. In this paper we focus our studies on the UW subset of the ATR line. The length of the transfer line presents challenges for tuning unto itself. The problem is further complicated by a 1.7 m vertical drop in order to get the beam from the AGS to RHIC.

The first part of the ATR (referred to as the U-line) starts with fast extraction from the AGS and stops before the vertical drop from the AGS to RHIC. The U-line consists of two bends. The first bend is 4.25°, consisting of two A-type dipole magnets. The second bend is an 8° bend consisting of four C-type combined-function magnets (placed in a FDDF arrangement), and thirteen quadrupoles. Optics in the U-line are configured to accomplish several goals. The Twiss parameters at the AGS extraction point must be matched, and provide achromatic transport of the beam to the exit of the 8° bend. The beam must be focused at the location of a thin gold foil which is placed just upstream of the quadrupole Q6 of the U-line. The Twiss parameters of the U-line must be matched to the ones at the origin of the W-line. Finally, the beam size should be kept small throughout to minimize losses.

The second part of the ATR (referred to as the W-line) introduces the vertical drop for injection into RHIC, and the matching sections for the injection lines. It contains eight C-type combined-function magnets that each make a of 2.5° bend, followed by six quadrupoles. The eight combined function magnets form a 20° achromatic horizontal bend placed in a (F-D) configuration. The W-Line is also responsible for lowering the beam elevation by 1.7 m. This is accomplished using two dipoles in an achromatic dogleg configuration. Along the line there are also a number of BPMs and correctors that are required to match the orbit of the beam into RHIC.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

FODO TEST PROBLEM

Before applying inverse models for anomaly detection on the full UW line, we first demonstrate the efficacy of this technique using a toy problem. Our toy problem is a FODO lattice, containing BPMs and correctors, in addition to the quadrupoles. We built a model that predicts the corrector settings for a given sequence of BPM settings. Because the quadrupoles supply a dipole kick when the beam does not traverse them on axis, any error in the predicted corrector setting should be strongly correlated to an error in the quadrupole strength. Here we utilized a neural network based inverse model training on simulation data collected using MAD-X. A schematic of the beam-line used for this study is shown in Fig. 1.

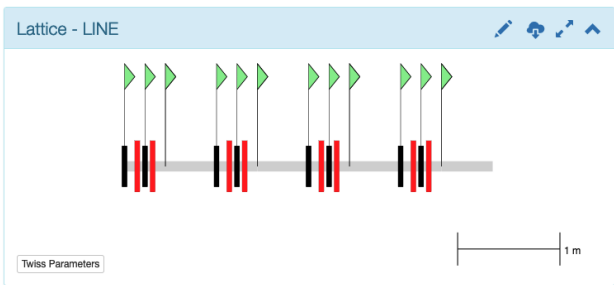


Figure 1: Screenshot of Sirepo visualization of toy lattice used for studying inverse models for anomaly detection. Quadrupoles are shown in red, correctors are black, and BPMs are indicated by green flags.

The beam line is composed of four identical FODO sequences. Each segment has two BPM/corrector pairs followed by a BPM after the second quadrupole. Having a large number of BPMs available, ensured that the inverse model was easy to train, which will allow us to use it to detect errors in the quadrupoles.

The training data consisted of 5000 examples simulated by randomly changing corrector strength and the initial beam position. The NN architecture was optimized as a function of the number of layers as well as the nodes per layer to improve training loss without over-fitting. Gaussian noise was used as a regularizer which leads to an increased noise on the validation loss as the epoch increases. Figure 2 shows the loss as a function of training epoch for the training and validation data.

Here we see that The training loss is a bit above the validation loss which indicates we could train longer. However, Fig. 3 shows the model prediction compared to the ground truth for each of the correctors (kickers). The relationship is almost perfectly linear in all cases, indicating the model is well trained.

Next we test the model using data with systematically introduced errors in individual quadrupoles. We ran the MAD-X simulations with random corrector strengths and initial beam positions. Then, the BPM outputs were used as inputs to the NN model — trained without quadrupole errors — to predict the corrector settings. These predicted corrector settings were then compared to the actual corrector settings.

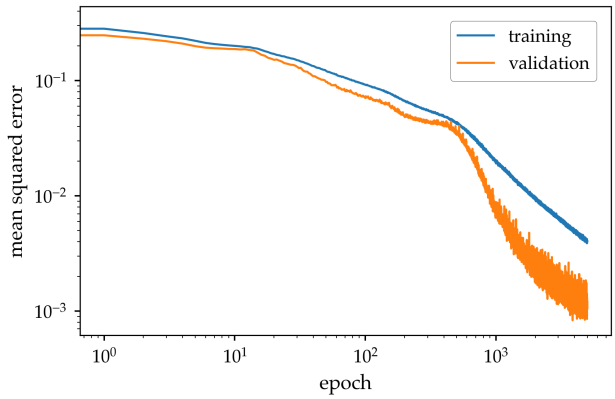


Figure 2: Mean squared error for the training and validation data as a function of the training epoch.

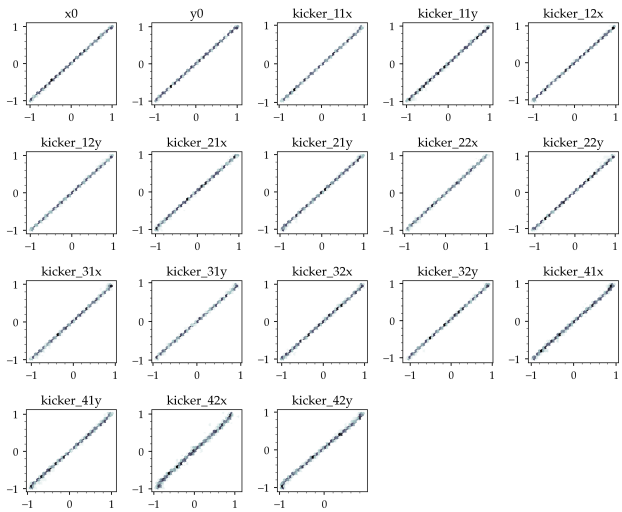


Figure 3: Predicted output vs. ground truth for the initial beam position and the kicker settings. The relationships are almost perfectly linear in all cases.

Figure 4 shows the predicted corrector setting versus the ground truth when testing the model using data generated with a strength error in one of the quadrupoles. Here it is very clear that the correctors in the first section of the beamline (kicker_11 and kicker_12) are not well reconstructed while all the other correctors are well reconstructed. This points to quadrupole strength errors in the first part of the beam-line.

We then studied the prediction error for a wide range of quadrupole errors and compared our ability to reconstruct the corrector settings. Figure 5 shows the normalized prediction error vs corrector index for data with different quadrupole errors. For each test case we introduced a single quadrupole error at different points along the beamline and we observed prediction errors in both the horizontal and vertical corrector settings.

Here we see that for a given quadupole there is a unique error signature for the correctors. Additionally, the corrector indices are arranged by their position in the beamline. As the quadropoles that are varied move down the beamline so do the errors in the corrector predictions. This is likely because

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

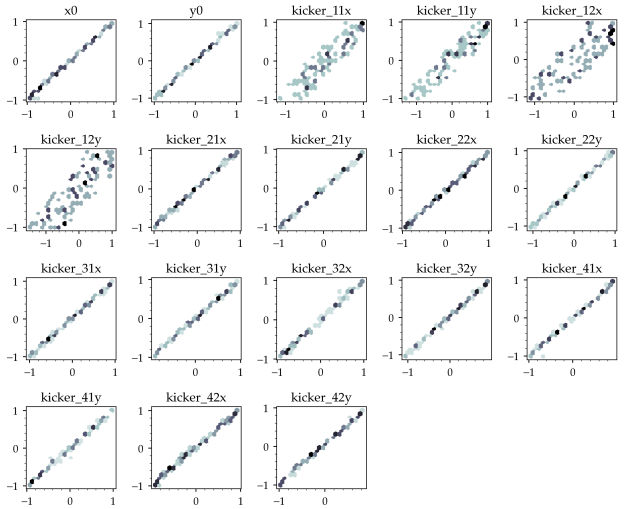


Figure 4: Predicted output vs. ground truth for data with a single quadrupole strength error.

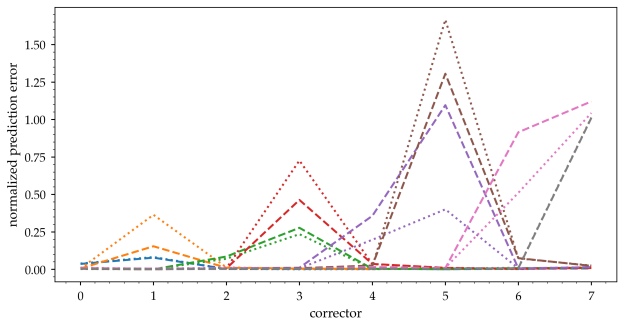


Figure 5: Normalized prediction error of corrector strengths given a range of Quadrupole errors. The dashed line is the horizontal corrector and the dotted line is the vertical corrector. The color indicates which Quadrupole was varied.

the BPM nearest the quadrupole with the error will be the most sensitive to this perturbation. Additionally due to the degenerate solutions in this type of model, it can be difficult to differentiate upstream errors from downstream errors. If we want to use this diagnostic for tuning, it is important to also examine the error in the corrector prediction as a function of the change in quadruple strength. Figure 6 shows the normalized prediction error as a function of quadrupole strength error for a single quadrupole.

There is a clear quadratic relationship between the quadrupole strength error and the normalized prediction error. This means that we can clearly detect anomalies in quadrupoles using our inverse model trained on the BPMs and correctors. Moreover, this means that we can vary the quads to minimize the model error based on measured BPMs and correctors, allowing us to tune the lattice without using destructive beam optics measurements.

ATR INVERSE MODEL

Given the success of this effort on the toy model we next applied this concept to the UW line in the ATR. The problem

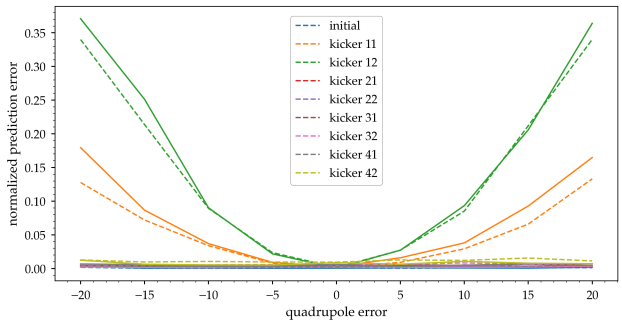


Figure 6: Normalized prediction error as a function of quadrupole strength error for each kicker (m^{-2}). The solid line is the horizontal kicker and the dashed line is the vertical kicker.

is a bit more complex as there are now not just quadrupoles, correctors, and BPMs, but also combined function bends and vertical bends.That said there are still comparatively few BPMs and correctors — only 26 and 14 respectively — making the model of similar scale to the toy problem. However there are 19 quadrupoles which significantly increases the complexity.

We trained the inverse model using 5000 samples, randomly varying the corrector strengths and beam initial positions. During our initial training of the inverse model four correctors (utv4, uth6, utv7, and wth1) were not well fit. This is likely due to the degenerate solutions that arise from the length of the transfer line. We attempted a wide range of solutions including sampling more data and working with image representations of the data 1-D convolutional layers. None of these methods were successful. When training inverse models there is always the possibility that the problem will not be fully invertable. In future work we will address this issue, however, for studies presented here we simply removed those four correctors from the prediction. Figure 7 shows the training and validation loss for the inverse model trained on the UW line.

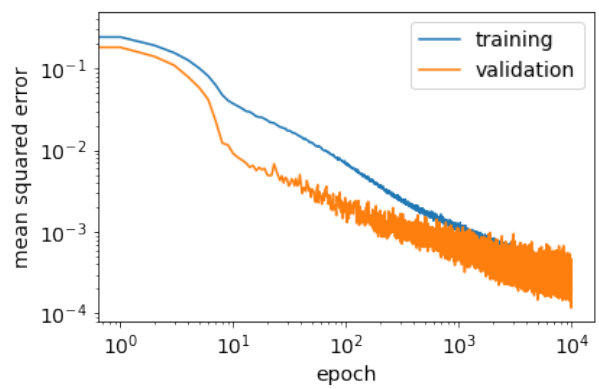


Figure 7: Training and validation loss as a function of training epoch for the UW line.

Here we see very good agreement between the training and validation set. Additionally the loss continues to decrease, showing that we are not over-fitting. For this study the data

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

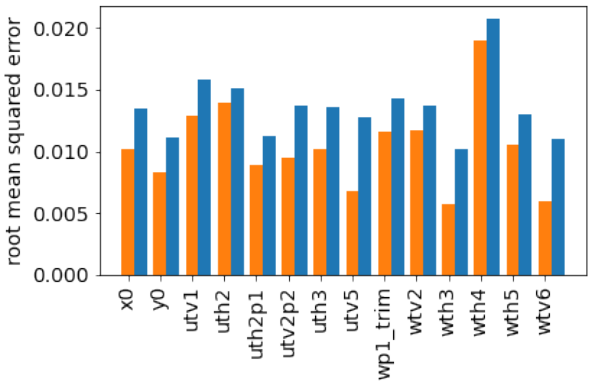


Figure 8: Root mean squared error on the training (blue) and validation (orange) datasets for the individual correctors.

were split into 80 % training and 20 % validation. Figure 8 shows the root mean squared error for each of the parameters individually.

The prediction error on the training and validation datasets are quite good. For this model we used 5 dense layers with 45 nodes each and Gaussian noise for regularization. The model used rectified linear units for the activation functions.

INVERSE MODEL ERROR STUDIES

With a trained inverse model we moved on to the error studies. Quadrupoles were individually varied over a range of ±20 % of their design excitation. We then simulated the UW line with random initial beam positions and random corrector settings. The BPM output from these simulations was used to predict the corrector settings using the model trained on data with the design quadrupole strengths. These simulations were run in two configurations, one where the initial positions were also varied randomly and one where the initial positions were not varied. The goal here was to understand if the model can differentiate between these two cases. During operations it is likely that the initial position will be relatively static.

Figure 9 shows the predicted corrector settings vs the ground truth for the validation set without quadrupole errors (black), the test set with a single quadrupole error and random initial position errors (red), and the test set with a single quadrupole error without initial position errors (blue).

Here we can clearly see that some correctors have a significant error in the prediction compared to others. For example Utv2p2 has a nice linear relationship but with a clear jump near zero. While UTH3, on the other hand, has a general increase in the error, as compared with WTV6, where the errors are relatively low. The model also shows a lack of ability to distinguish between cases where the initial position is varying and is not varying. The spread in the errors appear consistent between the two data sets. Indeed, Figs. 10 and 11 — which show the sensitivity of each corrector prediction to a particular quadrupole error — demonstrate that there is negligible difference between the cases with and without initial position errors. Here sensitivity is defined as the model

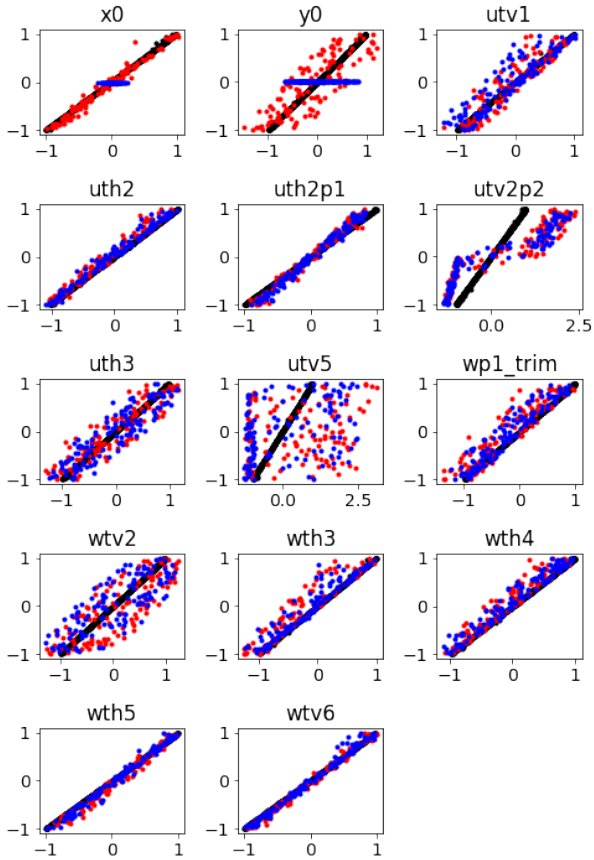


Figure 9: Predicted corrector settings as a function of the ground truth for, the three test cases. Without quadrupole errors: black. A single quadrupole error and random initial position errors: red. A single quadrupole error without initial position errors: blue.

error divided by the fractional change in quadrupole strength. These figures also show that there is a unique signature for each quadrupole and that the model clearly identifies errors in these magnets without any explicit knowledge of their existence.

These results are promising for the adaption of this method on the machine. Especially because the UW line is designed to be linwar, we expect the methods developed in simulation to behave similarly when transferred to the machine.

MACHINE STUDIES AND BPM INVERSE MODELING

We are in the early stages of testing our method on the machine, and have collected BPM and corrector data for the nominal machine configuration. Working with the UW line allows us to take data between injections to RHIC, parasitic to operations. These studies took approximately four hours of beam time without any interruption to operations.

Our main goals with this first study is to answer two key questions 1) how much data do we need to train an inverse model for the transfer line and 2) establish the feasibility of a neural network based inverse model for detecting quadrupole

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

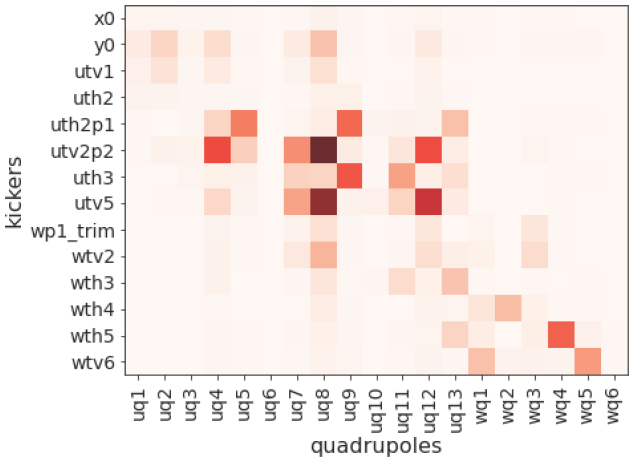


Figure 10: Sensitivity of each corrector prediction to individual quadruple errors. In this case the initial position was randomly varied along with the correctors.

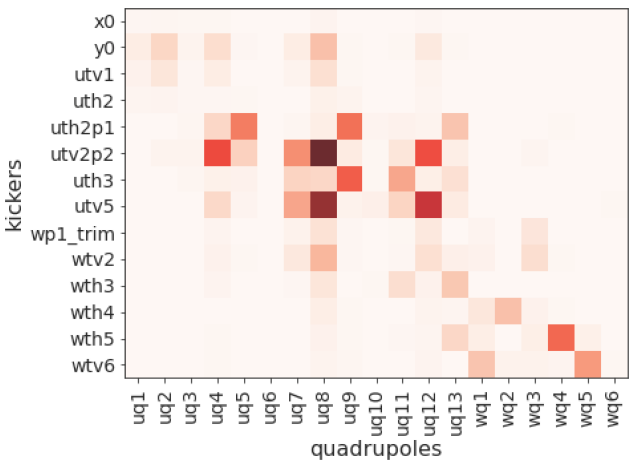


Figure 11: Sensitivity of each corrector prediction to individual quadruple errors. In this case the initial position not varied.

errors in the ATR line. With these data we can evaluate the accuracy of the ATR line MAD-X model, study the ability to transfer our NN inverse model from simulation data to measured data, and finally test our ML models on real data from the machine.

Figure 12 shows histograms of the beam position monitor data collected on the ATR line. Note that our initial dataset is relatively small, and has regions where there are quite a few outliers. The outliers in the dataset make it difficult to effectively sample the space for training neural networks. This will present a challenge when training on machine data alone. We plan to overcome this by combining the measured data with simulation data prior to testing on the machine.

With this dataset we performed some initial studies in order to understand the limitations of the measurements, and to help identify areas for additional data collection, or where simulations would be particularly useful. We are focused on using inverse models that predict the corrector settings from the BPM readings.

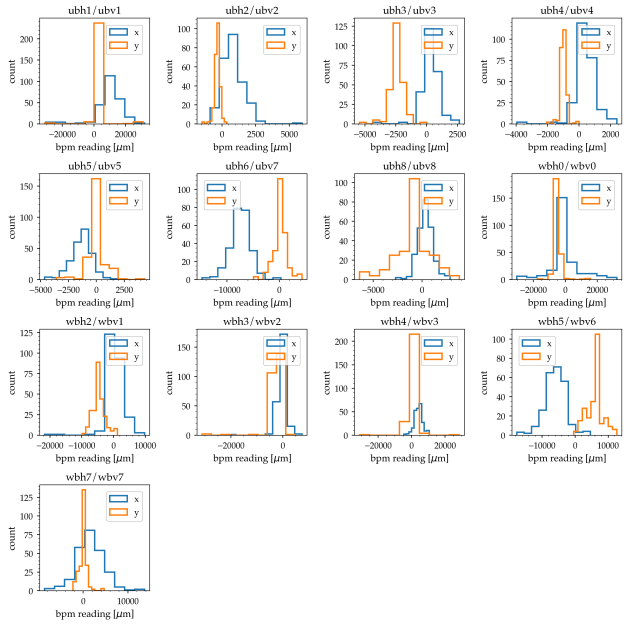


Figure 12: Histogram of BPM data collected from the ATR line. Horizontal (x) and vertical measurements are shown in blue and orange respectively.

For this study we varied numerous hyper parameters for the model, including: the number of nodes per layer, the number of layers, and the batch size. We also varied the regularization parameters to include both L2 regularization and Gaussian Noise. L2 regularization seeks to optimize the neural network architecture while simultaneously eliminating weights that are very small. The result is a more sparse network, but fewer nodes that are tracking noise in the data. Gaussian noise can assist the optimizer in getting out of local minima leading to a more robust solution. In the end, we trained on a batch size of 5 and aggressive Gaussian noise which helped the model fit such a small dataset. Figure 13 shows the training and validation loss as a function of training epoch for our neural network.

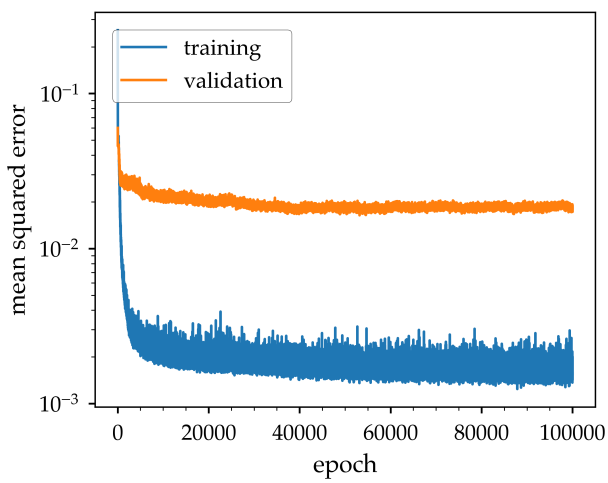


Figure 13: Loss curve for training and validation sets for the BPM data

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The training and validation loss both decrease quite rapidly and then flatten out. The validation loss continues to improve slightly but does not change significantly after about 40000 epochs. With small datasets it is important to train for a long time in addition to restarting with different batch sizes to introduce perturbations to the network that allow it to generalize better. The large offset between the training and validation loss indicates that the network would benefit from more examples. The fact that the validation loss is not increasing indicates that we are not overfitting though. In spite of the relatively large discrepancy between the training and validation loss, the performance on the validation set is quite reasonable. Figure 14 shows the predicted corrector setting compared to the ground truth for the validation set.

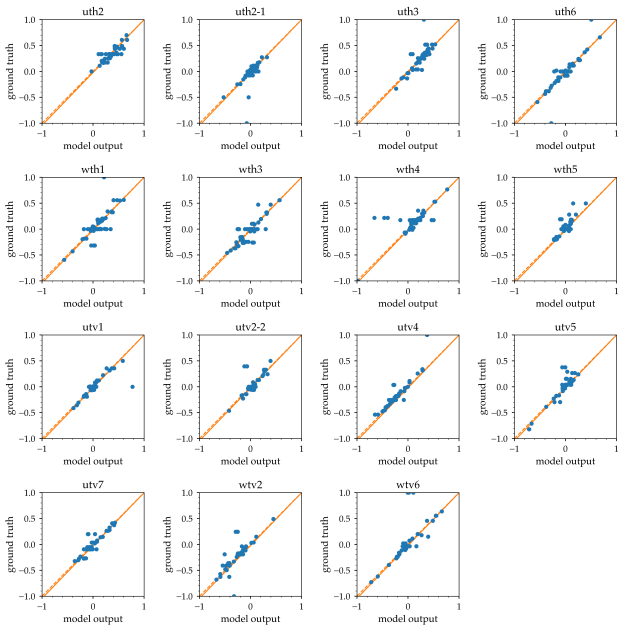


Figure 14: Fit results of the validation data for neural network trained on bpm on our study data

The solid orange line is the linear fit between the ground truth and the model output and the dashed line is the ideal fit should the model accurately reconstruct the corrector settings from the BPMs. In general the model performs relatively well, however as noted earlier it is clearly biased by small samples at extremes.

CONCLUSION

In this paper we explore the use of inverse models to detect errors in quadrupole strengths using BPM and corrector data. We have demonstrated an initial test using a toy model consisting of four FODO cells. We then scaled this to the Uw line on the ATR beamline at RHIC. Our results show that inverse models can identify quadrupole errors by comparing the predicted corrector setting to actual corrector settings. We also show that the each quadruple strength error yields a unique signature. We have begun transferring our work for testing on the machine and have shown a neural network inverse model can be trained on real BPM data from the UW line.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics, Award Number DE-SC0019682.

REFERENCES

[1] A.L. Edelen, S.G. Biedron, B.E. Chase, D. Edstrom, S.V. Milton, and P. Stabile, "Neural Networks for Modeling and Control of Particle Accelerators", in *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 878–897, April 2016. doi:10.1109/TNS.2016.2543203

[2] A.L. Edelen *et al.*, "Opportunities in Machine Learning for Particle Accelerators", 2018. arXiv:1811.03172[physics.acc-ph]

[3] C. Emma *et al.*, "Machine learning-based longitudinal phase space prediction of particle accelerators", *Phys. Rev. Accel. Beams*, vol. 21, p. 112802, 2018. doi:10.1103/PhysRevAccelBeams.21.112802

[4] A.L. Edelen, S. Biedron, J.P. Edelen, and S.V. Milton, "First Steps Toward Incorporating Image Based Diagnostics into Particle Accelerator Control Systems Using Convolutional Neural Networks", in *Proc. NAPAC'16*, Chicago, IL, USA, Oct. 2016, pp. 390–393. doi:10.18429/JACoW-NAPAC2016-TUPOA51

[5] A.L. Edelen *et al.*, "Using Neural Network Control Policies For Rapid Switching Between Beam Parameters in a Free Electron Laser", in *Proceedings of the 2017 Deep Learning for Physical Sciences Workshop at the 31st NeurIPS*. https://ml4physicalsciences.github.io/2017/files/nips_dlps_2017_16.pdf

[6] J.P. Edelen, K.A. Brown, N.M. Cook, and P.S. Dyer, "Optimal Control for Rapid Switching of Beam Energies for the ATR Line at BNL", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 789–794. doi:10.18429/JACoW-ICALEPCS2019-TUCPL07

[7] A. Scheinker *et al.*, "Demonstration of Model-Independent Control of the Longitudinal Phase Space of Electron Beams in the Linac-Coherent Light Source with Femtosecond Resolution", *Phys. Rev. Lett.*, vol. 121, p. 044801, 2018. doi:10.1103/PhysRevLett.121.044801

[8] A.L. Edelen *et al.*, "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems", *Phys. Rev. Accel. Beams*, vol. 23, p. 044601, 2020. doi:10.1103/PhysRevAccelBeams.23.044601

[9] J.P. Edelen and C.C. Hall, "Autoencoder Based Analysis of RF Parameters in the Fermilab Low Energy Linac", *Information*, vol. 12, no. 6, p. 238, 2021. doi:10.3390/info12060238

[10] *Proceedings of the 2021 Improving Scientific Software Conference*, W. Hu, D. Del Vento, and S. Su, Eds. Boulder, Co, USA: NCAR, 2021. doi:10.26024/p6mv-en77

[11] W.W. MacKay *et al.*, "AGS to RHIC transfer line: Design and commissioning", *Proc. EPAC'96*, Sitges, Spain, June 1996, paper MOP005G, pp. 2376–2378, 1996. https://jacow.org/e96/PAPERS/MOPG/MOP005G.PDF

[12] T. Satogata *et al.*, "Physics of the AGS-to-RHIC transfer line commissioning", in *Proc. EPAC'96*, Sitges, Spain, June 1996, paper MOP006G, pp. 2379–2381, 1996. https://jacow.org/e96/PAPERS/MOPG/MOP006G.PDF

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR01

MINT, AN ITER TOOL FOR INTERACTIVE VISUALIZATION OF DATA

L. Abadie, G. Carannante, I. Nunes, J. Panchumarti, S. D. Pinches, S. Simrock, M. Tsalas
ITER Organization, Saint-Paul Lez Durance, France
D. Makowski, P. Mazur, P. Perek, Lodz University of Technology Department of Microelectronics and Computer Science Wolczanska, Lodz, Poland
A. Neto, Fusion For Energy, Barcelona, Spain
S. S. Kalsi, Tata Consultancy Services, Pune, India

Abstract

ITER will produce large volumes of data that will need to be visualized and analyzed. This paper describes the development of a graphical data visualization and exploration tool, MINT (Make Informative and Nice Trends), for plant engineers, operators and physicists. It describes the early development phase from requirements capture to first release covering the mistakes, lessons learnt and future steps. The requirements were collected by interviewing the various stakeholders. The initial neglect of the architecture and user-friendliness turned out to be key points when developing such a tool for a project with a long lifetime like ITER. A modular architecture and clear definition of generic interfaces (abstraction layer) is crucial for such a long lifetime project and provides a robust basis for future adaptations to new plotting, processing and GUI libraries. The MINT application is based on an independent plotting library, which acts as a wrapper to the choice of underlying graphical libraries. This allows scientists and engineers to develop their own specific tools, which are immune to changes of the underlying graphical library. Data selection and retrieval have also been developed as a separate module with a well-defined data object interface to allow easy integration of additional data sources. The processing layer is also a separate module, which supports algebraic and user-defined functions. The development is based on Python [1] and uses Qt5 [2] as the visual backend. A first release of the 1-D trend tool (MINT) has already started and will be used for the ECH (Electron Cyclotron Heating) system commissioning. Other visualization tools will be developed in the future that build upon the same underlying modules.

INTRODUCTION

ITER is already producing data which need to be plotted and analyzed quickly. This paper describes the requirements and challenges, the development phases and various lessons learnt.

REQUIREMENTS

The top level requirement for data visualization is to be able to plot data for a time range or a pulse identifier.

Stakeholders

To define the detailed requirements, the first thing was to identify the main stakeholders. We identified three basic categories of users:

- Plant engineers whose main objective is to make system investigations; some of them will also perform research activities.
- The science team whose main objective is to analyze pulses and carry out research.
- The operation team whose primary focus will be the analysis of pulses.

Types of Data

The next step was to list the different data types of interest. After interviewing the stakeholders, the following list was constructed:

- Time and profile traces;
- Spectra;
- Fluxes (see Figure 1);
- Images and videos (see Figure 2)

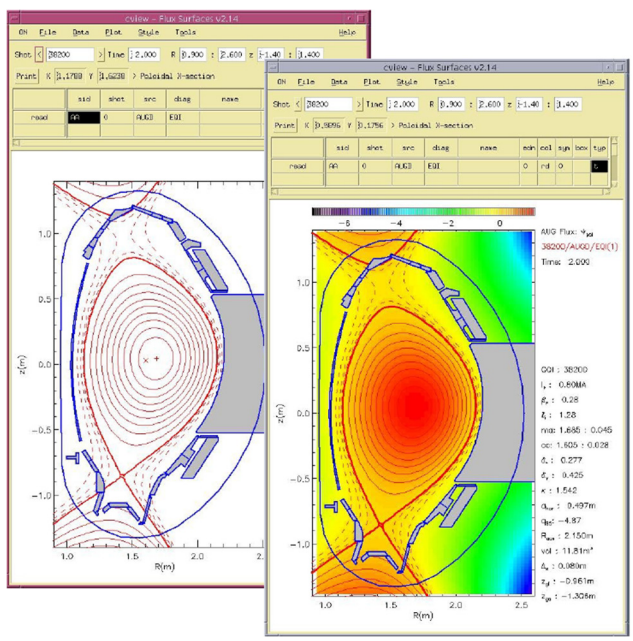


Figure 1: Example of magnetics flux surfaces as represented in CView used at ASDEX (courtesy from G.Conway).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR01

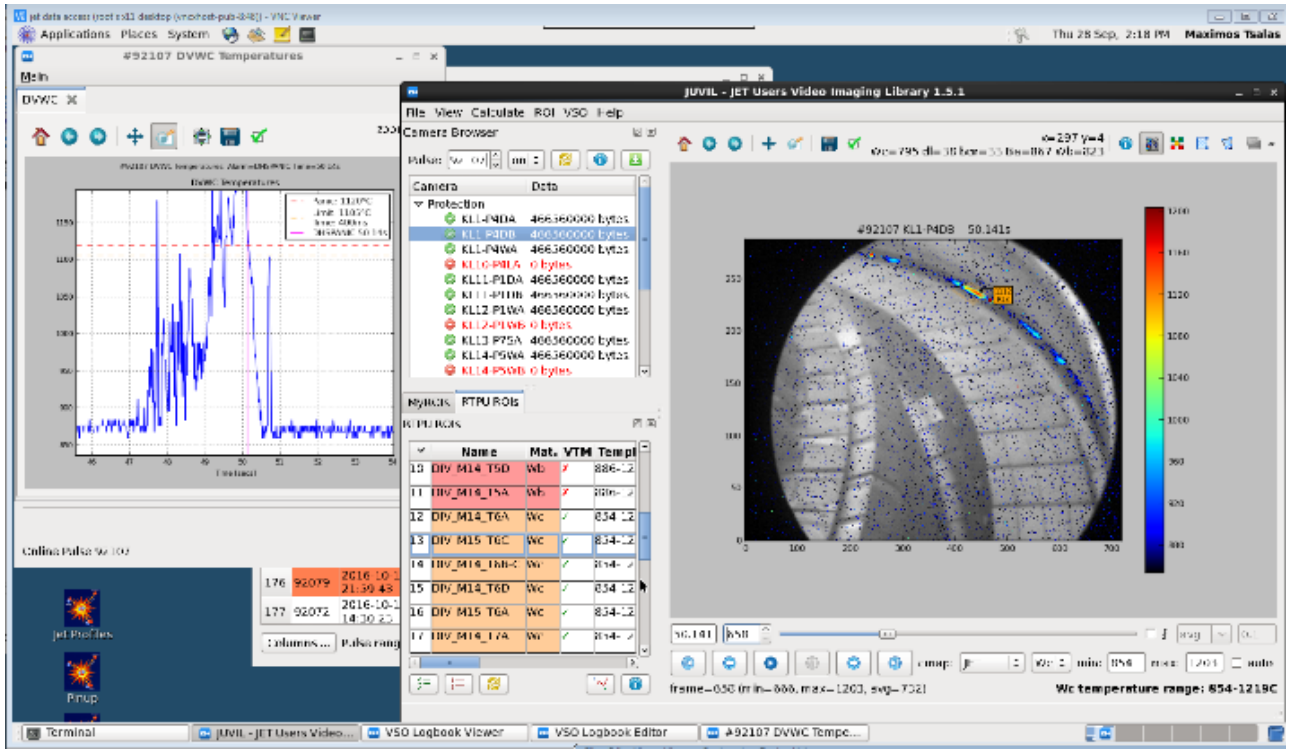


Figure 2: Example of image analysis (JUVIL used in JET).

Collecting Requirements

In order to get more practical users' feedback, a prototype was developed based on existing similar tools. It enabled collecting more precise user requirements. By making various demonstrations to the stakeholders a set of requirements were collected, including:

- Be able to overlay data from multiple pulses in one plot
- Be able to plot quickly 100 signals in multiple plots
- Be able to update the colors, to mark the points, to control how two points are connected
- Zoom/pan interactively: ITER has very fast sampled signals (kB-GB/sec). Loading all the points in memory on the client side would be very resource intensive. So there is a need to be able to dynamically retrieve data during zooming/panning operations with acceptable performance
- Be able to apply basic processing on one or more signals

DEVELOPMENT – PHASE I

Agile Development

As stated in the previous section, an existing product based on C++/Qt5 was initially extended. An agile development approach was opted to add new features. Bugzilla was used to track all new features, SVN to store the code, and Jenkins as the continuous integration system. Resources (CPU and memory) and performance optimization were the main objectives besides feature implementation. Thanks to ECH and ICH team, we could simulate the acquisition of fast signals from 2MHz to 125MHz. The demonstration of zooming interactively a signal sampled at 125MHz was successful: the zooming was done until one was able to measure the distance between 2 points (8 ns). Figure 3 shows the layout of our prototype.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR01

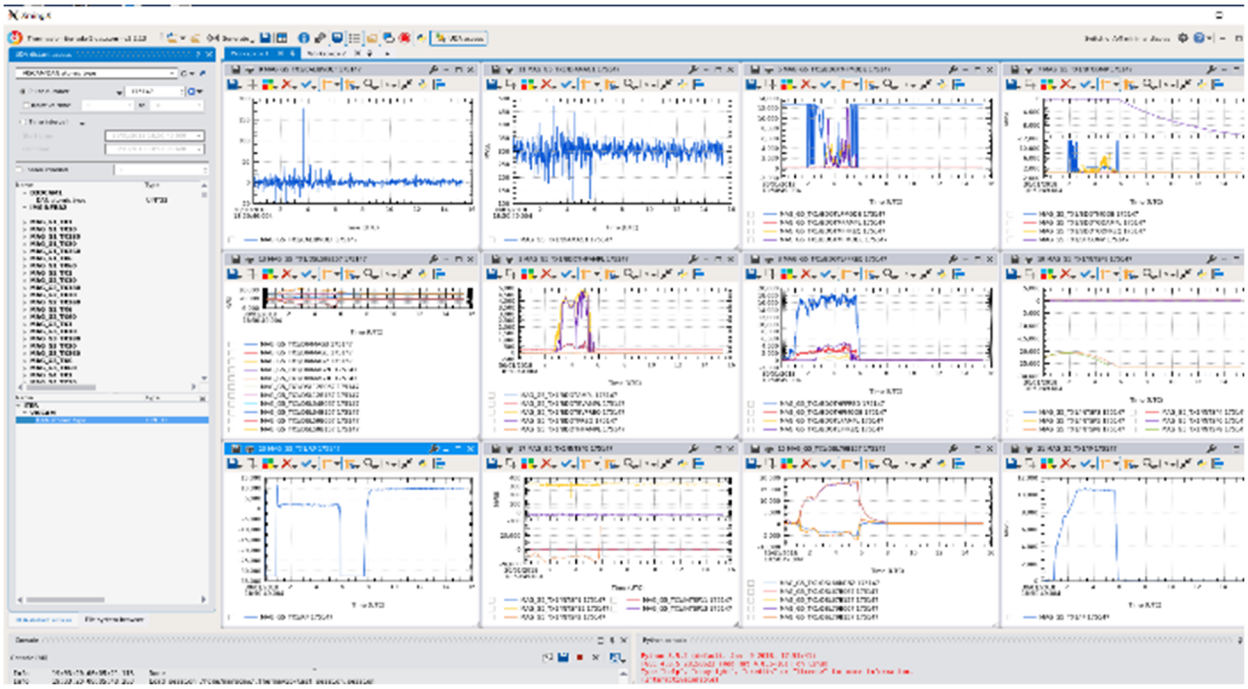


Figure 3: Look and feel of our first prototype based on Thermavip (tool used at CEA).

Training Stakeholders

After several months of development, a training was given to the stakeholders through dedicated separate sessions. They were provided the tool and requested to plot data for a particular time range and for a particular pulse identifier.

The results were not so good. Many had difficulties to execute basic actions. None of them could plot data without our help.

Scientists also challenged the tool by wanting to add new data sources for data retrieval, something that was not easy to do as per the implementation of the existing tool. As a consequence, the chosen approach was reflected upon in detail and the way forward discussed.

Lessons Learnt

During this initial development phase, it was realized that three key aspects were omitted:

1) Usability of the tool: non-developers shall be able to use the tool with very limited knowledge. When adding new features, the GUI became too complex and the main functionalities which needed to be kept simple, were not clearly identified.
2) Architecture: As a consequence of adopting an agile development approach, not enough thought was placed on the architecture, something that became clear when users requested to integrate a new data sources.
3) One big tool versus many tools: the initial concept was to have one big tool which could handle any data type and all stakeholders' views. Indeed it seemed attractive to provide users with one tool and then depending on their needs they can plot fluxes or profiles or time traces. With hindsight, it turned out that the GUI was too complex in terms of usability and maintainability. A decision was thus taken to develop tools with a limited but well defined scope to ensure user-friendliness and improved maintainability.

With all this in mind, a new development strategy was implemented.

DEVELOPMENT – PHASE II

Review of Existing Tokamak Tools

It was decided to invest more time analyzing existing tools currently in use on other tokamaks and stellarators.

It was found that the visualization tools on most devices shared similarities:

- Multiple tools are often developed to cover different aspects of data visualization: one tool for time traces and limited support for images, surfaces; one tool for fluxes and another tool for image analysis
- The User Interface for the time traces is based on a table and not a tree: it allows to be agnostic to the data structure used for the experimental signals and also eases the configuration of the plots and their signals
- Plotting data versus a pulse or a time range, zooming/panning, displaying crosshair were very straight forward
- Importance of good default settings

Architecture

As a result of the above review, a modular approach was subsequently adopted and it was furthermore decided to develop a plotting library so that scientists could reuse this layer to develop their own expert applications. Python and

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PySide [3] where selected as the basis for the development, Python being a commonly used language for scientific visualization.

The current architecture is based on 4 blocks:

- **Logging** : module which allows logging to file/console
- **DataAccess**: module which allows retrieving data from various data sources. To add a new data source, it is sufficient to add a new Python class and implement method to retrieve data. The data object used in this class is generic.
- **Processing** module: this module is responsible for processing the data as required after it has been fetched. It is responsible for time-base alignment when an operation involves more than one signal. Note that the processing is reapplied in case of a zoom/pan operation.
- **Iplotlib**: module which is responsible to plot the data and react to events such as zooming/panning. To demonstrate that the interface is independent from the graphics backend (here Matplotlib [4]), a VTK [5] back-end has also been developed. This allows a user to choose to use iplotlib either with Matplotlib (default) or VTK.
- **M.I.N.T.** (Make Informative and Nice Trends) is a tool that makes use of all the above modules. It is the primary tool for displaying time traces and analysis.

Current Status

Many features have been developed, as can be seen in Figure 4, such as plotting data for a given time range, or for a pulse identifier.

Features
View one or several 1-D trends (X axis is the time)
Support for pulse-based and time range data access (all the plots show the same time range/pulse)
Support for time range or pulse specialization at row level (allow comparing two plots side by side)
Support for pulse overlay
Support for interactive zoom
Support for crosshair (over multiple plots)
Support for envelope display
Support for pan interactive
Support for preferences update {color of a signal,draw-style {interpolated,stair}, display points with markers}
Add a plot title
Support for automatic X-axis scale (if I zoom or pan on one plot, all the plots will be rescaled to the same time window).
Import/Export list of variables
Save/load configuration a plot
Support for streaming
Support for full-mode
Support for basic processing of one signal
Support for Y and X-axis label edition
Generate a screenshot given a preference file in an headless environment (to allow report automation)

Figure 4: List of main features which have been actually implemented.

The tool layout with both back-ends is shown in Fig. 5 and Fig. 6.

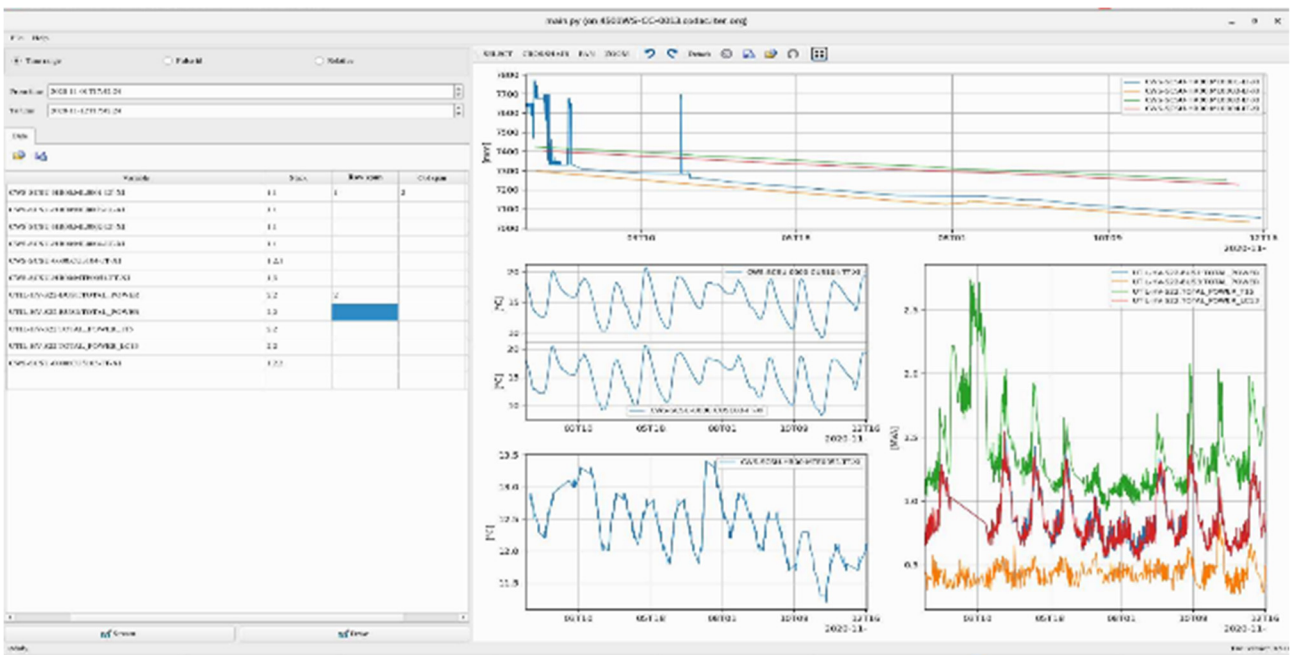


Figure 5: Layout of MINT (using Matplotib back-end).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR01

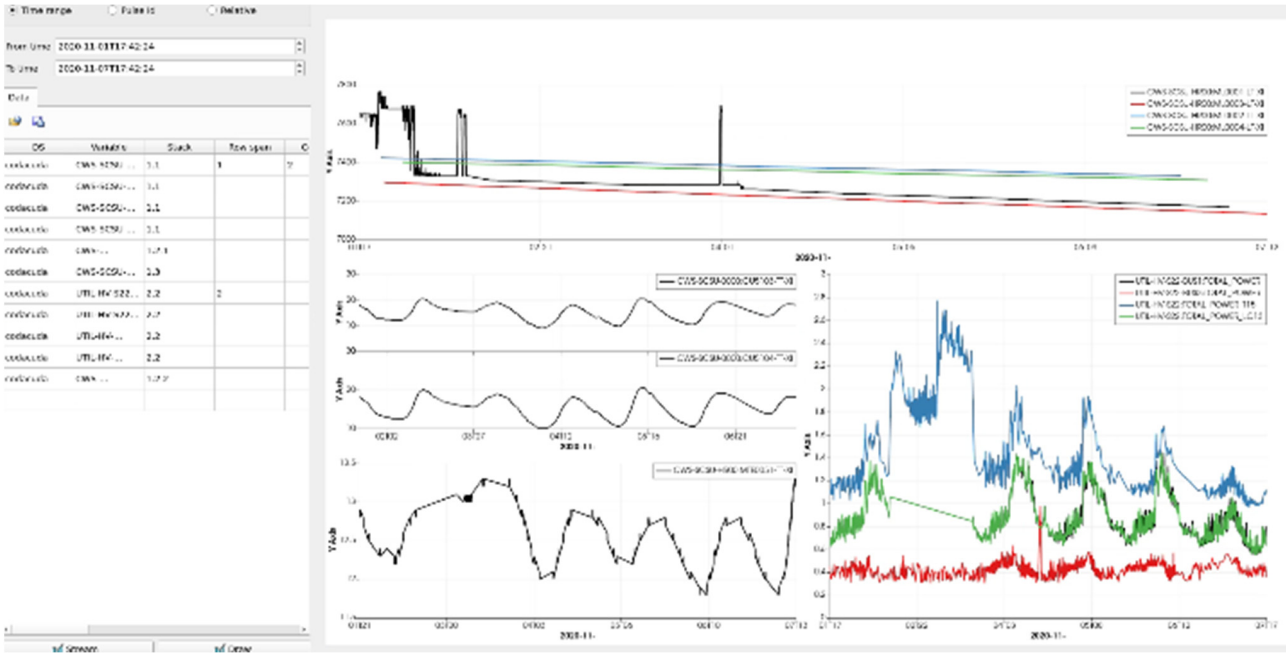


Figure 6: Layout of MINT (using VTK back-end).

Next Steps

There are still important features to be implemented including:

- Extending the data processing to support the manipulation of multiple signals and support for user-defined functions
- Extending the iplotlib module to support new plot types such as surfaces, a 2-D slice of 3-D data chosen with a slider, histograms, etc.
- Implement a browsing plugin to search for variables and metadata to ease filling the table.

It is also important to note that the performance for this tool will be critical. If zooming is too slow, users will give up and seek alternatives. That's why one of the next big investments of effort will be to improve the data access layer.

CONCLUSION

Good progress has been made on the development of a tool for data visualization. A lot has been learnt from the initial development experience and together with a more rigorous capture of the requirement this has provided a robust basis for subsequent development. The first version of the tool is now being actively used by the plant engineers (Cooling Water).

ACKNOWLEDGEMENTS

We would like to thank all stakeholders Trevor Blackman, Laura De Frutos, Peter Des Vries, Fabienne Kazarian, Arthur Leveque, Victor Moncada, Marco Riva, Marc-Henri Sautenet, Mireille Schneider, Luca Zabeo and the management Mikyung Park, Anders Wallander, and Tim Luce for their support in this work.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

[1] Python, https://www.python.org/

[2] Qt5, https://www.qt.io/home

[3] PySide, https://pypi.org/project/PySide2/

[4] Matplotlib, https://matplotlib.org/

[5] VTK, https://vtk.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR03

AUTOMATED SCHEDULER SOFTWARE BASED ON METRO UI DESIGN FOR MACE TELESCOPE

Mahesh Punna, Sandhya Mohanan, Padmini Sridharan, Pradeep Chandra, Sagar Vijay Godambe
BARC, Mumbai, India

Abstract

MACE Scheduler software generates automated schedule for the observations of preloaded high energy gamma-ray sources. The paper presents the design of MACE Scheduler software covering; source rise/set time calculation algorithms; auto and manual schedule generation; various data visualizations provided for schedule and source visibility reports. The schedule generation for a specific period is automated using a filter workflow. The sources are selected for scheduling by processing the sources through a series of customizable user defined filters; source visibility filter, priority filter, priority resolution filter. The workflow provides flexibility to apply any user tailored filter criteria that can be loaded dynamically using XML schema. Loosely coupled design allowed decoupling the astronomical timing calculation algorithms from schedule preparation workflow. Scheduler provides metro UI based interface for source filtering workflow generating auto-schedule, updating the generated schedules. Tree-map visualization helped to represent hierarchical multi-dimensional schedule information for the selected date range. WPF flat UI control templates focused more on content than chrome.

INTRODUCTION

Major Atmospheric Cherenkov Experiments (MACE) Telescope (Fig. 1) is a very high energy gamma-ray telescope set up by BARC, at Hanle, Ladakh, India for the study of gamma-ray emissions from various cosmic sources in the energy region of 20GeV-10TeV. The MACE Telescope Control System is a distributed control system which comprises of functionally diverse subsystems like Drive Control system, for moving the telescope towards specific sources at proper orientation; Mirror Alignment system, to focus the Cherenkov light onto the camera; CAMERA system, which recognizes the onset of cosmic events of interest and records the signal acquired. It also consists of Sky Monitoring system, which quantifies the sky transparency level and checks the tracking accuracy of the telescope during observation; Data Archiving System (DAS), and Weather Monitoring system (WMS).

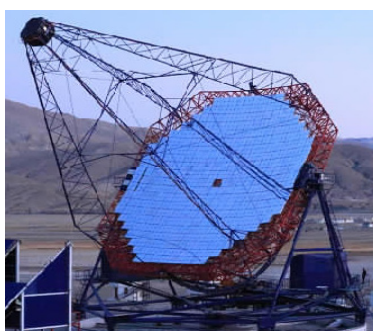


Figure 1: MACE Telescope.

The MACE Operator console holds the responsibility for integrated control and monitoring of these subsystems to conduct a successful observation run. Prior to the experiment an observation schedule selecting astronomical sources to be observed and the configuration of the various subsystems for the experiment is prepared by Astrophysicists.

The start-up sequence from the Operator console powers on the various subsystems in the required sequence and initialise the system and experiment configurations in each subsystem. This is done by running a sequence of commands (Fig. 2) from MACE Operator Console (OC), to bring the different subsystems to a ready state from where data acquisition can be started. These operation sequences are categorized into Initial run, Observation run, and Shut down run having associated pre-setup actions, experiment configuration, data acquisition and final shut down procedures respectively [1, 2]. The time required to carry out the initial run is accounted (X hours). The system automatically completes these activities and the Operator console starts the experiment X hours before the first observation.

OBS13102017_2	MU-LEONIS	9.88081	25.9998
Subsystem	Command	Status	
CAMERA	LoadExpConfig	Success	
CAMERA	ApplyHV	Success	
LCS	Load_Config	Success	
LCS	Get_Config	Success	
DAS	SetFilingParam	Success	
DAS	StartDAQ_DAS	Success	
SKMS	Load_Obs	Success	
TCU	StartTrack	Success	
SKMS	Start_Obs	Success	

Figure 2: Observation Command Sequence.

MACE SCHEDULER

The successful operation of the MACE Telescope for observing various high energy gamma celestial source by recording the Cherenkov events produced, highly depends on the schedule.

MACE Scheduler provides an interface to the Astrophysicists to generate observation schedule, which is required to streamline the observations of various sources with MACE telescope. A typical schedule contains 5-10 sources It is the scheduler's task to create a realisable schedule that will then be used in the observations. The challenge is to select one of those possible schedules, which will lead to good results. The schedule file thus created consists of a set of different sources (astronomical) and their coordinates along with the time of observation. Scheduler software generates date-wise schedules and stores in a configured centralized location.

MACE OC loads the schedule file for the current date and picks one source at a time to conduct the Observation Run, wherein respective source co-ordinates are sent to the Drive Control System for positioning the telescope towards the source and data acquisition is done as per the time spec-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR03

ified in the schedule. The schedule also chooses the experiment configuration to be loaded for each of the observation run.

Design Basis

The scheduler software is designed to provide the following functionalities:

- It should provide facility to select the schedule duration, which can be planned for day, lunation, up to a year or any specified period
- It should provide default source list with default Azimuth and Zenith coordinates
- It should have a provision for manual schedule generation for the selected input configuration
- It should have a provision for selecting the visible astronomical sources to be observed
- The user should be able to automate the schedule preparation by using a filter set for selecting the sources
- It should provide various timing plots for Sun, Twilight, Moon Rise set times for deciding the dark period (the time span during which, there will be no background illumination from sun, moon, twilight) for observation
- It should provide report generation facility.

Software Architecture

Scheduler software has been developed following layered architecture as shown in Fig. 3. Presentation layer handles user interface and implements custom controls, calendar controls, and tree map data visualization control. Application layer implements scheduler core modules like dark-period calculation and auto-scheduling controller and manual scheduling controllers. Algorithm layer implements rise and set time calculation algorithms for various celestial objects. Each layer defines an interface for the services it requires from the lower layer. The main aim of layered design [3] is the separation of concerns among the components. Application layer modules do not depend on lower layer concrete modules, instead they depend on the abstractions following Dependency Inversion principle. This helps to build loosely coupled software.

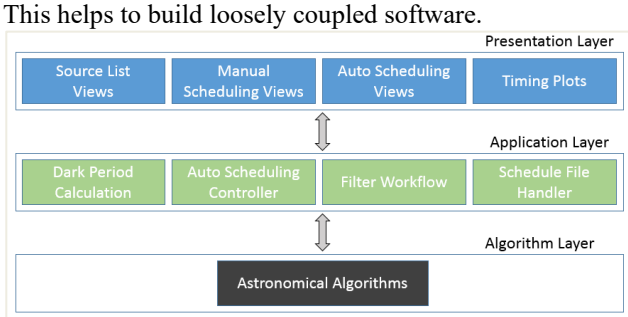


Figure 3: Software Architecture.

SOFTWARE DESIGN

Algorithms

The software implements the following astronomical algorithms [4, 5]:

- Calculating Rise/Set time for Sun, Moon and any custom source with defined coordinates
- Calculating Moon Phase
- Calculating the dark period for any day
- Calculating Azimuth and Zenith, given source coordinates and location

Astronomical algorithms are implemented following Adapter design pattern [6] (Fig. 4) and defines an algorithm interface for the application and makes it compatible with various algorithm implementations.

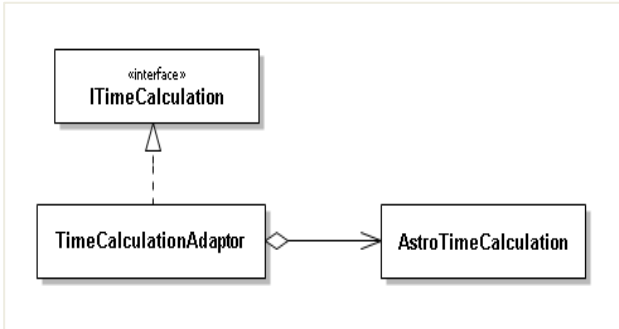


Figure 4: Adaptor Design Pattern for Astronomical Algorithms.

Source List

The source list is organized into catalogues of various types (Optical, Gamma-ray etc.) and each source catalogue contains various source list with positional information (Right Ascension (RA) and Declination (DEC)) and priority. The source list data is stored in user configurable csv format. Source view provides the selected sources visibility information, polar view for Azimuth trend and Zenith trend plot for 24 hours of the selected day (Fig. 5). Selected source zenith trend plotted along with dark period helps to decide the observation period for the day.

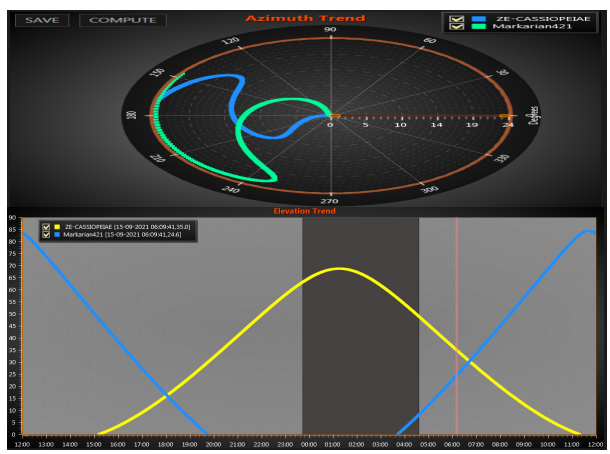


Figure 5: Selected source's azimuth and elevation trend along with dark period.

Schedule Generation

The main responsibility of MACE Scheduler software is to enable the user to generate observation schedules for the

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

selected period and selected location. The software provides two modes for schedule generation; Auto, manual modes.

Auto-Schedule Generation

Observation Schedule is automated by filtering and ordering the sources based on the user-defined criteria and scheduling the ordered sources as per the dark period window. Each source is configured to have a default scheduling period. Dark time available for the selected date is allotted to the ordered sources as per the source configured period.

The following are the steps for the auto-schedule generation.

1. **Input** Location coordinates (Latitude & Longitude), time zone, date selection, observation duration and source list are taken as input for auto-schedule generation
2. **Sources Selection** The sources are grouped into 4 categories; Optical (OP), Galactic (GA), Extra-Galactic (EG) and Positional (PO). The required source group or all groups or any user interested sources can be selected for scheduling
3. **Source Filtering** The sources are filtered and ordered from all the selected source list. To enable the user to control how the sources are filtered, a source reduction workflow has been created. The workflow projects intermediate filter results, enabling the user to see each filter output.

FilterContainer (Fig. 6) acts a Filter Factory that creates Filter Objects and builds filter workflow. Strategy Design Pattern helped to load the selected filters dynamically. The design adapts to new custom filters.

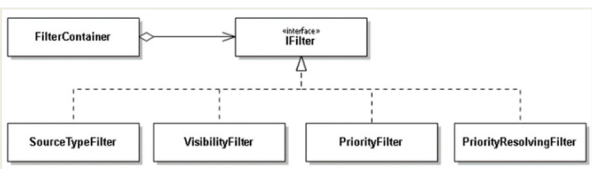


Figure 6: Filter Container.

The workflow consists of the following filters. Each filter takes input from previous filter and produces source list for the next filter input

- **Source Type Filter** The selected source list can further be filtered based on the source type (OP, GA, EG and PO)
- **Visibility Filter** Source observation is normally carried out during dark period. Hence the source should be visible within the dark period for the selected day. Rise/set time of previously filtered sources is calculated and checked if the source visible period overlaps with the dark time span. Thus, visibility filter produces a set of sources that can be observed during dark period.
- **Priority Filter** Each source is assigned a user-defined priority. Previously filtered sources will be ordered based on the priority. User can change the priority while building the workflow.
- **Priority Resolver** The same priority sources in the ordered source list can be resolved using a priority resolved criteria. Currently used criteria for resolving the priority is set time. The source that sets first is given more priority
- **Custom user filter** The workflow provides flexibility to apply any user tailored filter criteria. The user can define any customized filter and load it dynamically through structured XML schema as shown in Fig. 7.

For example, consider the following filter criteria: Source Azimuth > 50 and having 45 < Zenith < 90

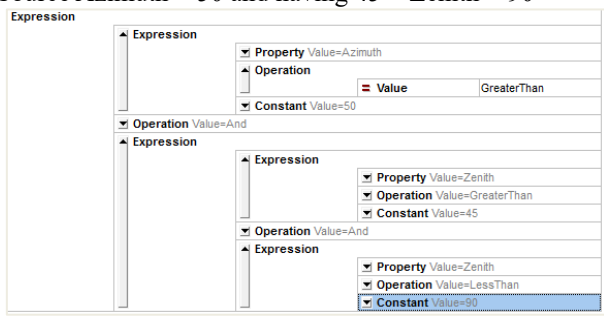


Figure 7: Custom Filter Expression.

Custom filters help to extend the software with various filter criteria depending on the requirements

- **Final Source List & Schedule Generation** The source list will be finally reduced to an ordered list for scheduling the observations depending on the Dark Period availability and a schedule for the selected day will be generated (Fig. 8)

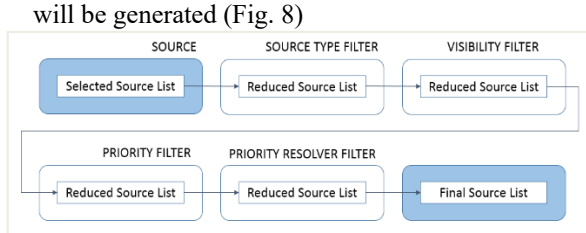


Figure 8: Filter Workflow.

Manual Schedule Generation

The software also provides manual mode for schedule generation. For the selected day, the visible sources having overlap with Dark Period are selected for scheduling. For the selected source, depending on the tracking mode (Wobble, On/Off), configuration is set. Rise Time plot (Fig. 9) displays Rise and Set timing for Sun, Twilight period, Moon and Dark Period. It also shows the selected source rise/set timings. The plot helps to properly schedule the sources as per the available dark time span. The software provides custom user control to select observation time span up to minute. Rise time plot shows scheduled/added observations

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR03

Figure 9: Rise set time plot.

The following Fig. 10 shows manual schedule generation UML sequence diagram [7].

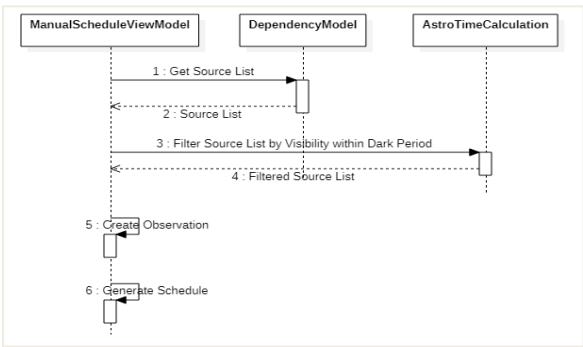


Figure 10: Manual Schedule Generation.

Reports

The software provides different views for visualizing Sun, Moon, selected source visibility and schedule data. Schedule and visibility data can be exported to readable txt/excel formats. Visibility report (Fig. 11) contains dark period for the day and selected sources visibility for the selected date range.

NAME	01-09-2021	02-09-2021	03-09-2021
BLACK PERIOD	20:03-00:48	20:01-01:41	20:00-02:40
AL-ANDROMEDAE	23:21-04:04	23:17-04:00	23:13-03:56
BE-CASSIOPEIAE	00:20-03:10	00:16-03:06	00:12-03:02

Figure 11: Visibility report.

A schedule file contains the following information; Sources being observed, observation time span, dark time available, tracking mode information and configuration to be used. Schedule reports contain day-wise scheduled sources with the observation time span. Schedule information is shown in matrix layout, where each row contains date-wise scheduled sources. However, the matrix data is difficult to comprehend (Fig. 12) as the date range and number of sources observed increases.

10-09-2021	OP- ZE-CYGNI	08:41 09:11	GA-0FGL1958.1+2848	09:11 09:41	EG-BLLacertae	09:41 10:11
11-09-2021	OP- ZE-CYGNI	09:18 09:48	GA-0FGL1958.1+2848	09:48 10:18	EG-BLLacertae	10:18 10:48
12-09-2021	OP- ZE-CYGNI	10:00 10:30	OP- ZE-PEGASI	10:30 11:00	EG-BLLacertae	11:00 11:30
13-09-2021	OP- ZE-CYGNI	10:48 11:18	OP- ZE-PEGASI	11:18 11:48	OP- BE-ANDROMEDAE	11:48 12:18
14-09-2021	OP- ZE-CYGNI	11:42 12:12				

Figure 12: Schedule Report.

To get better insight into the scheduling statistics tree map control (Fig. 13) has been developed. Tree map view [8] helps to flatten the hierarchical information and presents multi-dimensional data. The view highlights the source has with highest scheduling period and positions it to the top left as indicated by the size of the block. For each source block, the date on which it got the highest schedule period is indicated by the size of date block. Color shows the dark period duration. The color intensity of a block is proportional to the dark period available to it.

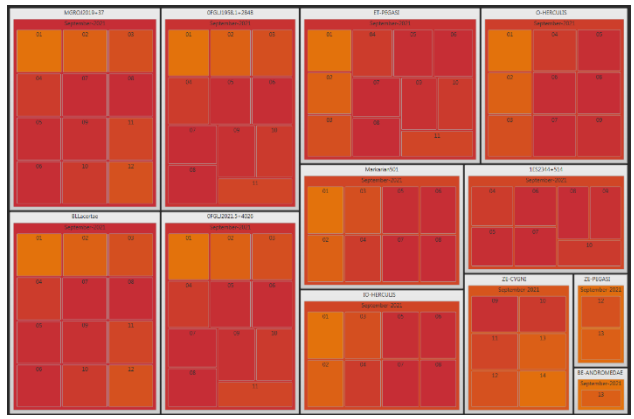


Figure 13: Schedule Info tree map visualization.

Metro/Flat UX [9]

Scheduler user interface design has been developed following simple visual appearance, allowing the user to focus on content rather than chrome (colours, shadows etc.). Interface/custom control design has been inspired by Microsoft Design Language (MDL). It is also referred as Flat Design, which follows minimalistic approach using simple shapes, colours and icons for creating UI controls that are informative, thus avoiding the UI overload that comes from the visual effects. Flat UI design has been followed throughout the scheduler software.

General Implementation

UI has been developed in WPF framework [10] and astronomy algorithms have been developed in C# following MVVM [11] architectural style, separating user interface logic from the application logic. Custom controls (ex: Custom Calendar control, auto schedule filter workflow control) are created by defining control templates to configure the control look and UX. Separating view logic from application logic helps to reduce coupling and improves application testability and maintainability. The following Fig. 14 shows the custom calendar control with orange markers to indicate if the schedule is prepared for that day.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

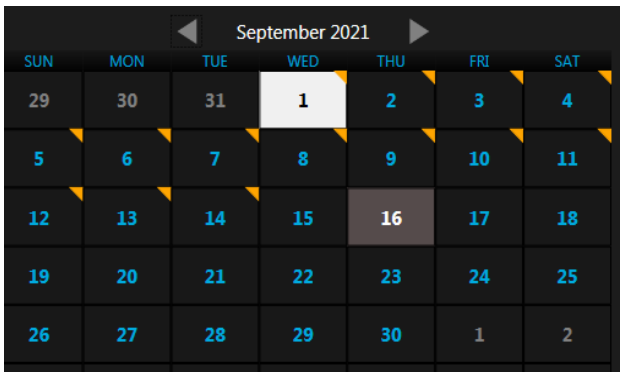


Figure 14: Custom calendar control.

CONCLUSION

MACE Scheduler can automate the schedule generation through filter workflow for the selected date range and location. The software provides intuitive user interface for scheduling manual observations. The software consists of timing plots showing the available Dark Period and Sun, Moon, Selected Source's rise/set times, allowing the user to take a quick decision on schedule duration and time. The software design allows the scheduler extend to any custom developed astronomy algorithms. The software provides reports view using matrix form and tree map visualization for showing multi-dimensional schedule information. Schedule and Visibility information can be exported to readable format. MACE telescope has become fully operational in 2020 and scheduler has been used for preparing observations

ACKNOWLEDGEMENTS

Authors would like to thank the colleagues from Astrophysical Sciences Division, BARC Mumbai for their suggestions and guidance.

REFERENCES

[1] S. Srivastava, A. Jain, P. M. Nair, P. Sridharan, "MACE camera controller embedded software: Redesign for robustness and maintainability", ELSEVIER Publications, Astronomy and Computing, Volume 30, Jaunary 2020, https://doi.org/10.1016/j.ascom.2019.100358

[2] Shikha Srivastava, Anushri Jain, Janhavi Deshpande, Mahesh Punna, Preetha M. Nair, Padmini S., "Design of Data Acquisition, Run-Control & Monitoring Software for MACE Telescope", National Symposium on Gamma Ray Astronomy-2020.

[3] Mark Richards (2015), "Software Architecture Patterns", O'Reilly Media, Inc, ISBN: 9781491924242.

[4] Duffett-Smith, P., "Practical Astronomy with your Calculator (3rd ed.)", Cambridge University Press, 1989. doi:10.1017/CBO9780511564895

[5] Jean Meeus, "Astronomical Algorithms (2nd ed.)", Willmann-Bell, 1998, ISBN-10:0943396611, ISBN-13:978-0943396613.

[6] John Gamma, Erich Helm, Richard Johnson, Ralph Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley, 1994, pp. 139ff. ISBN 0-201-63361-2.

[7] www.uml-diagrams.org/sequence-diagrams.html

[8] https://en.wikipedia.org/wiki/Treemapping

[9] https://www.microsoft.com/design/fluent/

[10] https://docs.microsoft.com/en-us/dotnet/desktop/wpf/introduction-to-wpf

[11] https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL01

CONTROL SYSTEM MANAGEMENT AND DEPLOYMENT AT MAX IV

B. Bertrand*, Á. Freitas, V. Hardion, MAX IV, Lund, Sweden

Abstract

The control systems of big research facilities like synchrotron are composed of many different hardware and software parts. Deploying and maintaining such systems require proper workflows and tools. MAX IV has been using Ansible to manage and deploy its full control system, both software and infrastructure, for quite some time with great success. All required software (i.e. tango devices, GUIs...) used to be packaged as RPMs (Red Hat Package Manager) making deployment and dependencies management easy. Using RPMs brings many advantages (big community, well tested packages, stability) but also comes with a few drawbacks, mainly the dependency to the release cycle of the Operating System. The Python ecosystem is changing quickly and using recent modules can become challenging with RPMs. We have been investigating conda as an alternative package manager. Conda is a popular open-source package, dependency and environment management system. This paper will describe our workflow and experience working with both package managers.

INTRODUCTION

The Controls & IT group, also called KITS, is responsible for the whole IT infrastructure at MAX IV. This includes everything from control system hardware and software to data storage, high performance computing, scientific software and information management systems. Within KITS, the Control System Software team manages all the software linked to the control system. With the accelerator and 16 beamlines, this represents more than 330 physical and virtual machines to configure and maintain. Ansible [1] was chosen for its simplicity of use as detailed in CONFIGURATION MANAGEMENT OF THE CONTROL SYSTEM [2] and is a great help to achieve this. The control system is made of many components that often have dependencies with each other: tango devices, controllers, GUIs. Building and being able to deploy each software individually without breaking another part is not straightforward. This requires some tools and is exactly why package managers were designed. One of their role is to keep track of dependencies between packages to ensure coherence and avoid conflicts. Using a package manager makes it easier to distribute, manage and update software.

PACKAGE MANAGEMENT

RPM

The RPM Package Manager [3] (RPM) is the package management system that runs on Red Hat Enterprise Linux, CentOS, and Fedora. As CentOS is the default Operating System at MAX IV, using RPM to distribute internal software was an obvious choice.

RPM gives us access to a large numbers of high quality packages from the main CentOS repository and others like EPEL [4], the Extra Packages for Enterprise Linux. This provides solid foundation to build on and is one huge advantage of Operating System package managers.

SPEC file RPM creation is usually based on a SPEC file [5]. It is the recipe that rpmbuild uses to build an RPM. It contains metadata like the name of the package, version, license, as well as the instructions to build the software from source with all the required dependencies as seen in Fig. 1.

```
Summary:    Tango device for Linkam T96 heater
Name:       tangods-linkamt96
Version:    1.2.0
Release:    1%{?dist}.maxlab
License:    GPL
URL:        http://www.maxiv.lu.se
Source:     %{name}-%{version}.tar.gz
Requires:   lib-maxiv-linkam-t96
Requires:   linkam-sdk
Requires:   lib-maxiv-common-cpp >= 4.0.0
Requires:   libtango9
BuildRequires: lib-maxiv-linkam-t96-devel
BuildRequires: linkam-sdk-devel
BuildRequires: lib-maxiv-common-cpp-devel >= 4.0.0
BuildRequires: libtango9-devel
# for pogo Makefile templates:
BuildRequires: tango-java

%description
Tango device for Linkam T96 heater

%prep
%setup -q

%build
make

%install
[ -z %{buildroot} ] || rm -rf %{buildroot}

# install bins
pushd bin > /dev/null
for f in *; do
    install -D -m755 $f %{buildroot}%{_bindir}/$f
done
popd > /dev/null

%files
%defattr (-,root,root,755)
%{_bindir}/*
```

Figure 1: RPM SPEC file (extract).

C++ projects are packaged using a SPEC file. RPM creation is handled by a GitLab CI [6] pipeline using maxpkg,

* benjamin.bertrand@maxiv.lu.se

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

a MAX IV plugin to rpkg [7] for managing RPM packaging from a git repository.

fpm Most software at MAX IV are written in Python. *setup.py*, even if not recommended anymore [8], was the historical way to package a Python library or application with *setuptools* and is still widely used today. fpm [9] is a command-line program designed to help build packages. It can take different source types (directory, rubygem, python package...) and convert them to a target type, most common being "rpm" and "deb". Using fpm avoids having to write a SPEC file to create RPM. fpm can take the metadata required to build a package from the *setup.py* file.

```
fpm -s python -t rpm \
  --python-bin python3.6 \
  --rpm-dist el7 \
  -p results \
  --python-package-name-prefix python36 \
  --python-setup-py-arguments=--prefix=/usr \
  --no-python-downcase-dependencies \
  ${FPM_FLAGS} \
  --url $CI_PROJECT_URL --verbose .
```

Figure 2: Example of `fpm` command.

Figure 2 shows a typical command to create an RPM from a Python repository. It is automatically run by our GitLab CI pipeline. fpm takes the dependencies defined by *install_requires* in the *setup.py* file and automatically adds the *python36* prefix to match the RPM naming convention. If a Python application had the following *install_requires=["taurus", "PyYAML", "pytango"]*, fpm would create a RPM with the dependencies python36-taurus, python36-PyYAML and python36-pytango. If there is a mismatch between the Python package and RPM name, it is possible to pass extra arguments using the *FPM_FLAGS* variable and overwrite the RPM dependencies, i.e. *FPM_FLAGS: '–no-python-dependencies -d python36-numpy,python36-dateutil'* could be used for a package depending on the *python-dateutil* library.

With the CentOS Project decision to shift its focus from CentOS Linux to CentOS Stream [10] and the uncertainty this created, migration to CentOS 8 was stopped. This forced us to remain on CentOS 7 and that started to create issues. CentOS is known for its stability, which is very important, but can become a problem when recent software are needed. Sardana [11] version 3 was very difficult to install as RPM due to the versions of the dependencies required. This is one of the reason we started to look at alternatives package managers like conda [12].

Conda

Conda is an open-source package, dependency and environment management for any language. It is cross-platform and runs on Windows, macOS and Linux. By using anaconda compilers, binaries created with conda-build can run on any modern Linux distribution. Being OS independent is an interesting feature that makes changing OS or migrating to a new one easier. With RPM, even when upgrading between two major releases of the same distribution, a rebuild of all packages is required. It's not the case with conda. The same package can be installed on CentOS 7 and 8 or even Debian.

Anaconda [13], the company behind conda provides some default channels with a large amount of packages. In the past years, conda-forge [14] became the de facto channel when using conda. Conda-forge is a community-led collection of recipes, build infrastructure and distributions for the conda package manager. It allows developers to automatically build recipe in a clean and repeatable way on Windows, Linux and macOS.

In 2021, the Tango community started to publish tango packages to conda-forge as mentioned in the THE TANGO CONTROLS COLLABORATION STATUS [15]. MAX IV also made available some packages that could be useful to the community like dsconfig or svgsynoptic2. Creating a recipe for conda-forge isn't very difficult but there is a review process done by volunteers that can take some time. For internal software we need a faster way to release packages and we deployed our own internal conda server based on Quetz [16].

Quetz is an open-source conda packages server. We use it to proxy external channels, like conda-forge, allowing conda packages to be installed without internet access. We also have some local channels to store packages created internally.

One complaint that people have about conda is that it can be slow. This is true when using large channels, and we even noticed a quite high memory usage (Fig. 3).

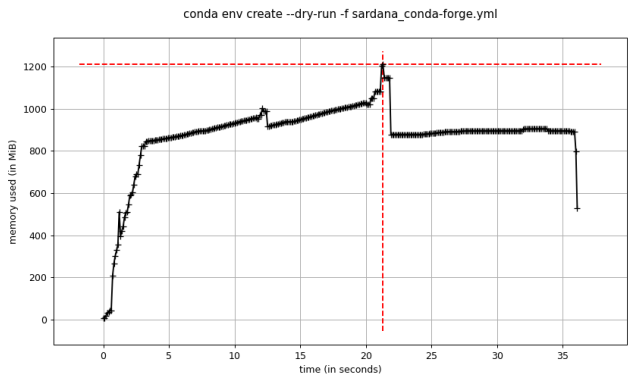


Figure 3: Sardana env creation with conda-forge.

The list of packages available in a channel is stored in a *repodata.json* file. It includes the description of all packages and their dependencies. That file needs to be downloaded and parsed to resolve an environment. The bigger that file, the more work for conda. As packages are never removed from a channel like conda-forge, the repodata.json file keeps growing. It is about 141MB today for the conda-forge linux-64 channel.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL01

One way to improve performances is mamba [17], a re-implementation of conda in C++ for maximum efficiency. It was conceived as a drop-in replacement for conda, using libsolv for much faster dependency solving. Mamba is robust and fast (Fig. 4) but not 100% compatible with conda yet, especially for conda-env commands, meaning we couldn't rely on it for all operations.

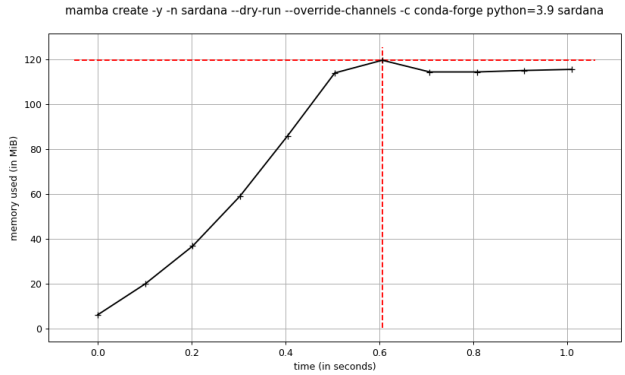


Figure 4: Sardana env creation with mamba.

As conda performances depend on the channel size, we created our own local *mini-conda-forge* channel, a subset of conda-forge. Having a channel with only the packages we are interested in gives a big boost in performances, both in time and memory usage (Fig. 5). Using that channel, instead of conda-forge, solving an environment with sardana decreases the time from over 35 seconds to 1.4 seconds and memory usage from 1.2GB to only 45MB! The result is almost identical to using mamba. Note that those figures don't take into account the download of the repodata file, that was already cached, nor the download and installation of the packages.

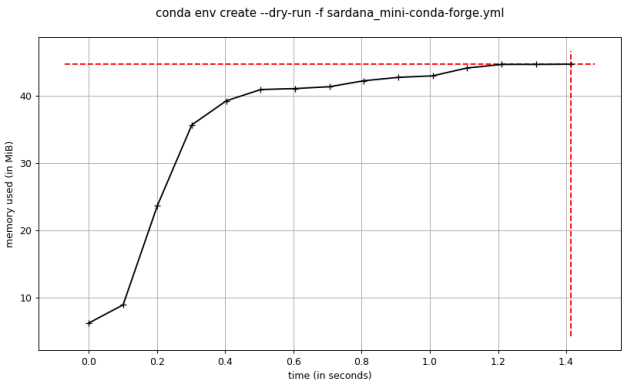


Figure 5: Sardana env creation with mini-conda-forge.

To keep this channel up-to-date automatically, a GitLab CI pipeline is run on schedule every night. It downloads new packages, and their dependencies, from conda-forge and uploads them to mini-conda-forge. The packages to download are based on a list of environments we want to be able to install, that is defined in a text file (Fig. 6). Making new packages available only requires to update this packages specs file.

```
# We want to be able to install conda and mamba
conda
mamba
# Default packages for a sardana environment
python=3.9 sardana bitshuffle matplotlib spyder pytango taurus taurus_pyqtgraph h5py hdf5plugin icepap sockio
# lavue
python=3.9 lavue pytango h5py bitshuffle hdf5plugin
# silx
python=3.9 silx h5py bitshuffle hdf5plugin
# pymca
python=3.9 pymca matplotlib pyqt
```

Figure 6: Example of mini-conda-forge environments list.

Building a conda package requires a recipe, which is defined in a *meta.yaml* file, with the information needed to create the package. This is equivalent to the SPEC file for RPM. Figure 7 is a typical example of a pure Python package on conda-forge.

```
{% set name = "dsconfig" %}
{% set version = "1.6.0" %}

package:
  name: {{ name|lower }}
  version: {{ version }}

source:
  url: https://pypi.io/packages/source/{{ name[0] }}/{{ name }}/dsconfig-{{ version }}.tar.gz
  sha256: e7709b0fa920b9a41460c0613ea2925aed089b8852cc8fa10ed2c3056ef82b56

build:
  number: 0
  noarch: python
  entry_points:
    - xls2json = dsconfig.excel:main
    - json2tango = dsconfig.json2tango:main
  script: {{ PYTHON }} -m pip install . -vv

requirements:
  host:
    - pip
    - python >=3.6
  run:
    - jsonpatch >=1.13
    - jsonschema
    - pytango
    - python >=3.6
    - six
    - xlrd

test:
  imports:
    - dsconfig
    - dsconfig.appending_dict
  commands:
    - pip check
    - xls2json --help
    - json2tango --help
  requires:
    - pip

about:
  home: https://gitlab.com/MaxIV/lib-maxiv-dsconfig
  summary: Library and utilities for Tango device configuration.
  license: GPL-3.0-or-later
  license_file: LICENSE.txt
  description: |
    This is a command line tool for managing configuration of Tango device servers.
    Tango-Controls is a software toolkit for building control systems.
  dev_url: https://gitlab.com/MaxIV/lib-maxiv-dsconfig

extra:
  recipe-maintainers:
    - beenje
```

Figure 7: `conda-forge` recipe.

To package software we develop internally we decided it was easier to make the conda recipe part of the source repository. This removes the need to have a second repository to maintain. Figure 8 shows an example.

conda-build provides a macro to parse the *setup.py* file. It can be used to get the version from that file as well as the runtime dependencies, making it easier to write and maintain the recipe.

Our GitLab CI pipeline template will automatically build and upload the conda package to our Quetz server if it detects a *recipe/meta.yaml* file in a repository.

ANSIBLE

Ansible was chosen early at MAX IV as the solution to deploy and manage the control system.

RPM Deployment

Ansible has a builtin *yum* module to manage packages with the *yum* package manager. This makes it easy to deploy RPM. For ease of use and consistency, a generic playbook was created to deploy internal software packages with RPM. The *packages_stable* and *packages_testing* variables define the list of packages to be deployed. Those variables are maintained in the Ansible inventory in the proper group or host variables file. The default version for each package is

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

```
{% set data = load_setup_py_data(setup_file="../setup.py",
  from_recipe_dir=True) %}

package:
  name: tango_exporter
  version: {{ data.get('version') }}

source:
  path: ..

build:
  number: 0
  noarch: python
  script: {{ PYTHON }} -m pip install . -vv
  entry_points:
    - tango_exporter = tango_exporter:main

requirements:
  host:
    - pip
    - python >=3.6
  run:
    - python >=3.6
{% for dep in data['install_requires'] %}
    - {{ dep.lower() }}
{% endfor %}

test:
  imports:
    - tango_exporter
  requires:
    - pip
  commands:
    - pip check
    - tango_exporter --help

about:
  home: https://gitlab.maxiv.lu.se/kits-maxiv/app-maxiv-tangoexporter
  license: GPL-3.0-or-later
  license_file: ../LICENSE.txt
  summary: Prometheus exporter for a Tango control system.
```

Figure 8: `local` recipe.

centralized in the *all* group for consistency (Fig. 9a). Developers are encouraged to use this default version but can also pin it if needed or even use latest (Fig. 9b).

```
versions:
  python-taurus: 4.5.1-7.el7
  python-dsconfig: 1.4.0-1.el7
  python-facadedevice: 1.0.3.dev1-1.el7
  python-fandango: 14.3.0-1.el7
  python36-sdm: 1.6.1
  tangods-pathfixer: 1.5.7
  python-pyicepap: 2.8.1-1.el7
  python36-sherlock: 1.0.4
```

(a) `versions` definition.

```
packages_stable:
  python-taurus: default
  python-dsconfig: default
  python-facadedevice: 0.9.0
  python-fandango: 10.9
  python36-sdm: default
  tangods-pathfixer: default
  python-pyicepap: default
  python36-sherlock: latest
```

(b) `packages` definition.

Figure 9: RPM packages - Ansible inventory.

Conda Deployment

Ansible doesn't have any builtin module to interact with conda. MAX IV has its own modules (Fig. 10) based on the ones developed by ESS [18].

- The *conda* module can install, update or remove conda packages. It works with a list of conda packages.
- The *conda_env* module manages environment using an *environment.yml* file.

Mamba is used by default by the *conda* module but it's currently not compatible with the *conda_env* one.

```
- name: install flask 2.0 and Python 3.9
  conda:
    name:
      - python=3.9
      - flask=2.0
    state: present
    environment: myapp
```

(a) `conda` module.

```
- name: create myenv environment
  conda_env:
    name: myenv
    state: present
    file: /tmp/environment.yml
```

(b) `conda_env` module.

Figure 10: Ansible modules

Our ans_maxiv_role_conda Ansible role installs and configures both conda and mamba. It can also be used to create a list of conda environments by setting the *conda_envs* variable in the inventory. Conda environments are isolated by nature. You usually have to activate one to use it. To be transparent for the users, the role can create wrappers for command line applications. The wrapper is a simple script that activates the environment and runs the command from the env. Wrappers are deployed under `/usr/local/bin`. Users don't have to know that the application they run is installed in a conda environment. Figs. 11 and 12 show how to define such an environment and the resulting wrapper.

```
conda_envs:
  - env_name: silx
    dependencies:
      - python=3.9
      - silx=0.15.1
    wrappers:
      - silx
```

Figure 11: `conda_envs` definition.

```
#!/bin/bash

# Some cli need the env to be activated
source /opt/conda/etc/profile.d/conda.sh
conda activate silx

/opt/conda/envs/silx/bin/silx "$@"
```

Figure 12: `/usr/local/bin/silx` wrapper.

Conda is also used to deploy Sardana version 3. A specific Ansible role and playbook were developed to deploy it based on an environment.yml file. This format allows to define both conda and Python packages (installed with pip). The yaml file is created from a template defined in the role and can be customized, to add extra packages, based on different variables in the inventory. Figure 13 shows an example of such a resulting environment. Wrappers are also installed under `/usr/local/bin` for all sardana commands to make them globally available.

CONCLUSION

Using a package manager to build, distribute and update software is a requirement in modern software development,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL01

```
channels: [maxiv-kits, mini-conda-forge]
dependencies:
- pytango=9.3.3
- taurus=4.7.1
- taurus_pyqtgraph
- h5py
- hdf5plugin
- icepap=3.6.2
- sockio=0.15.0
- taurusgui-maxpeemenergy=0.2.2
- python=3.9
- sardana=3.1.2
- bitshuffle
- matplotlib
- spyder
- pytest
- pip
- pip:
    - --trusted-host repo.maxiv.lu.se
    - '-i http://repo.maxiv.lu.se/devpi/maxiv/prod'
    - taurusgui-scangui3==1.1.3
    - taurusgui-quickgui3==2.1.3
name: sardana
```

Figure 13: Sardana environment.

as git is for version control. There are different kinds of package managers that all have their strengths and weaknesses. RPM is by nature integrated with the OS. It's very stable and can be used to create systemd services for example. Conda is OS independent and creates isolated environments, avoiding the risk of dependencies conflicts, and giving more freedom in the packages that can be installed. Ansible, combined with RPM or conda, gives us a reliable and reproducible way to deploy and maintain the control system.

REFERENCES

[1] Ansible, https://docs.ansible.com/ansible/latest/index.html

[2] V. Hardion *et al.*, "Configuration Management of the Control System", in *Proc. ICALPECS'13*, San Francisco, CA, USA, 2013, paper THPPC013.

[3] RPM Package Manager, https://rpm.org

[4] EPEL, https://docs.fedoraproject.org/en-US/epel/

[5] SPEC file, https://rpm-packaging-guide.github.io/#what-is-a-spec-file

[6] GitLab CI/CD, https://docs.gitlab.com/ee/ci/

[7] rpkg, https://docs.pagure.org/rpkg/index.html

[8] Packaging Python Projects, https://packaging.python.org/tutorials/packaging-projects/#configuring-metadata

[9] fpm, https://fpm.readthedocs.io

[10] CentOS Stream, https://blog.centos.org/2020/12/future-is-centos-stream/

[11] Sardana, https://sardana-controls.org

[12] Conda, https://conda.io

[13] Anaconda, https://www.anaconda.com

[14] Conda-Forge Community, "The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem", *Zenodo*, 2015. doi:10.5281/zenodo.4774216

[15] A. Götz *et al.*, "The Tango Controls Collaboration Status in 2021", presented at ICALPECS 2021, Shanghai, China, 2021, paper WEAR01, this conference.

[16] Quetz, https://quetz.readthedocs.io

[17] Mamba, https://mamba.readthedocs.io

[18] ESS conda Ansible modules, https://gitlab.esss.lu.se/ics-ansible-galaxy/ics-ans-role-conda

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

EXPLORING ALTERNATIVES AND DESIGNING THE NEXT GENERATION OF REAL-TIME CONTROL SYSTEM FOR ASTRONOMICAL OBSERVATORIES

T. C. Shen*, J. Sepulveda, ALMA Obervatory, Santiago, Chile
P. Galeas, F. Huenupan, S. Carrasco, R. Augsburger, R. Seguel, U. de La Frontera, Temuco, Chile

Abstract

The ALMA Observatory was inaugurated in 2013; after the first eight years of successful operation, obsolescence emerged in different areas. One of the most critical areas is the control bus of the hardware devices located in antennas, based on a customized version of CAN bus. Initial studies were performed to explore alternatives, and one of the candidates can be a solution based on EtherCAT technology. This paper compares the current architecture with a new proposal compatible with the existing hardware devices, providing the foundation for new subsystems associated with ALMA 2030 initiatives. The progress of a proof of concept is reported, which explores the possibility of embedding the existing ALMA monitor and control data structure into EtherCAT frames, using EtherCAT as the primary communication protocol to monitor and control hardware devices of ALMA telescope subsystems.

INTRODUCTION

The ALMA Observatory was inaugurated in 2013; after the first eight years of successful operation, obsolescence emerged in different areas. One of the most critical areas is the control bus of the hardware devices located in antennas, based on a customized version of CAN bus. Initial studies were performed to explore alternatives, and one of the candidates can be a solution based on EtherCAT [1] technology. This paper compares the current architecture with a new proposal not only compatible with the existing hardware devices, but also provides the foundation for new subsystems associated with ALMA 2030 initiatives. This document reports the progress achieved in a proof of concept that explores the possibility to embed the ALMA monitor & control protocol into a EtherCAT protocol. The main goal of this phase is to obtain the technical assessment of the feasibility to implement the EtherCAT as the communication protocol to monitor and control hardware devices/controller in all the subsystems that comprises the ALMA telescope. This is a collaboration project between ALMA Observatory and the Universidad de La Frontera in the context of a QUIMAL fund (QUIMAL190009) [2], which is sponsored by Chilean National Agency for Research and Development (ANID). The main objective is to design, implement and evaluate an possible alternative of the existing antenna real time computer.

* tshen@alma.cl

THE ALMA DISTRIBUTED CONTROL SYSTEM

The ALMA control software is a distributed system that is divided into several subsystems, each focusing on different stages of the observation process. The subsystems provide software interfaces to transfer communication messages in a coordinated manner between them. Likewise, the control system is also responsible for coordinating, by means of events and/or command messages, all the activities involved in the different observation steps.

The main subsystems are Control, Correlator, Scheduling, Telescope Calibration, Executive and Archive. The software for each subsystem is implemented in one (or more) programming languages (C++, Java, Python) that support the ALMA common software (ACS) [3], CORBA-based framework. The official operating system is Red Hat Enterprise Linux Server release 7.6.

The Antenna Bus Master (ABM)

The ABM is a dedicated real-time computer to monitor and control the antenna hardware devices. The purpose of this computer is to process low level messages from all antenna devices, using a particular implementation of the CAN communication protocol [4]. The scheme of monitor and control conducted by the ABM computer is accomplished with adoption of the ALMA Monitor Bus (AMB) specification. It is a particular ALMA protocol, based on a CAN bus, to communicate with hardware elements, which defines a unique master connected with several slaves in the same bus. The AMB specification promotes a configuration that converts the transaction of CAN messages in a deterministic communication of the command control messages. The master, in a timely manner, is the only agent on the bus that can sends messages and wait responses of the other elements involved in the CAN bus. Similarly, the ABM make uses of five independent ALMA Monitor Bus channels to communicate with the devices spread out in the antenna. In every channel the ABM real-time computer acts as the CAN master and antenna hardware devices are the slaves on the CAN bus.

Hardware Device Interface (AMBSI)

The ALMA Monitor and Control Bus Interface (AMBSI) is a standard interface that defines, on the one side the physical connection of nodes on the bus, and on the other side, the application level protocol that nodes must conform to be monitored and controlled by a software control system. The AMBSI specification outlines that each ALMA device has

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

a node address (0-2030), a unique 64 bit serial number, the ALMA M&C bus (AMB) is a master/slave multi-drop serial bus. The ALMA specification is implemented in two standard interfaces, known as the ALMA Monitor and Control Standard Interface 1 & 2 (AMBSI1, AMBSI2). In addition, the standard also established that the low level protocol is Controller Area Network (CAN) serial bus operated in a master/slave mode by a dedicated bus master. The remote frame transmission request (RTR) is not used by the bus master to gather monitor data.

Timing Event

The ALMA software infrastructure synchronizes the activities with the antennas by a common time base [5]. The time base definition is based on an electronic pulse signal with a period of 48 milliseconds, denominated as Time Event (TE) and absolute Gregorian time associated with each TE. The time system is provided by group of specialized hardware elements and software component denominated the Array Time. It is the main reference time for all machines connected in the ALMA system environment. This is a special coordination mechanism of the time that enables interactions between participants, in a time synchronized manner, according to a unique universal time provided by this dedicated time scheme. The coordination between different participants is performed by means of a time events.

The activities of monitor/control of hardware devices are completed in synchronization with the Time Events (TE). The Fig. 1 below shows a TE-related control/monitor command, the processing of this type control command may begin at the starts of every TE pulse. Any slave (hardware device) that receives a control command from the CAN master (ABM) must processes it in the first 24 ms of TE. In the same way, monitor commands are processed in a second window of 20 milliseconds of the 40 ms TE-window.

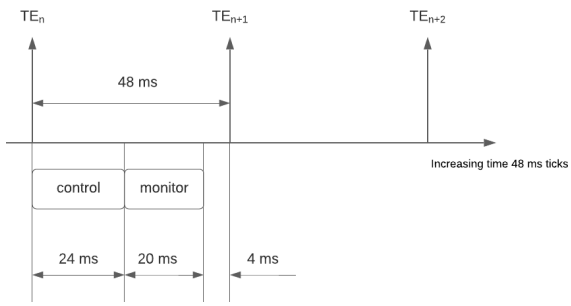


Figure 1: Time Event linked with control and monitor commands.

THE PROPOSED DESIGN

The design take into account different parameters of open standards in control bus. The Table 1 shows the attributes of EtherCAT, CAN-Bus (CANOpen), RS-232, RS485 bus standards.

Different scenarios were analyzed for the observatory in the coming years, and these are the most relevant requirements and uses cases in the new control system:

- monitor and control of the existing hardware devices of the ALMA telescope.
- monitor and control of new hardware devices to be introduced mainly due to obsolescence problems.
- monitor and control of new hardware devices to be introduced by improvement and automation in the antenna remote recovery activities.
- monitor and control of new hardware devices of the ALMA 2030 developments.

Given the aforementioned scenarios, it was decided to explore a solution that could combine the EtherCAT, CANOpen and OPC UA protocols. The new design must be capable to combine, in the distributed control system, the different flavors of protocols and interfaces of existing design and also guarantee the future extension of the this new design. The resulting architecture must minimize the impact in the higher control software layer based on ACS. Additionally, the new design should provide a way to allow hardware engineers to be able to access the control bus for maintenance purposes, without the need to have the entire stack of control software up and running. This is an important aspect, because it prevent hardware and software engineers to work in parallel in different areas during the scarcely available technical time.

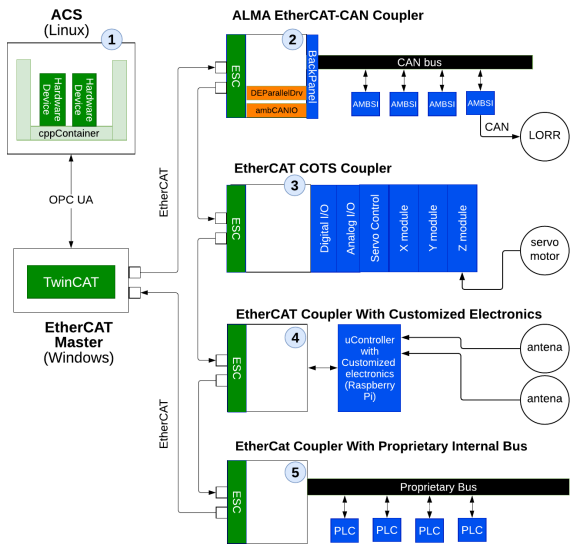


Figure 2: Control bus use cases in the coming years in ALMA.

The scenario or use cases displayed in the Fig. 2 are the following:

ALMA Device Controller

The use case 1: The existing ABM's functionalities are separated into two parts. The first part represents the user-space (no real-time) to be executed in a Linux server with ACS. The existing "Hardware Device" components are adapted to use a OPC UA client driver, which sends (via Ethernet) messages to the second part (real-time) of the proposed system. Finally, an application must be developed to run within the IPC on top of TwinCAT 3, in order to verify

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 1: Control Bus Comparison

	EtherCAT	CAN-Bus	RS232	RS485
Speed/Cable Length	100 Mbps at 100 m between two nodes for full speed	1 Mbps at 40 m from the start to end nodes	19,200 bps at 15 m from the start to end nodes	1 Mbps at 120 m from the start to end nodes
Cable Topology	Daisy Chain, Point-to-Point, Combination of the above	Daisy Chain	Point-to-Point	Daisy Chain, Point-to-Point
Max Number of Nodes	65535	128	1	32
Open Source Software	Yes	Most of CANOpen is proprietary	No predetermined protocol	No predetermined protocol
Required Hardware at master	Regular Ethernet adapter	PCI, USB, IC-type interface	PCI, USB, IC-type interface	PCI, USB, IC-type interface

the reception of the OPC UA messages and dispatch the message to the EtherCAT bus.

ALMA Current CAN Bus Device

The use case 2: This is the most important scenario for ALMA, because it tackles the current obsolescence problem of ABM real-time computers. In this scenario, the ACS components deployed in containers of the Linux Server running ACS will be able to communicate via EtherCAT with hardware devices connected to the CAN bus. The commands sent by hardware device component are processed by the OPC UA server and translated into EtherCAT frames, once the message is received by an EtherCAT-CAN coupler, it will converts the messages into the CAN messages compatible with the current implementation of ALMA control bus.

COTS EtherCAT Coupler with Digital, Analog I/O

The use case 3: This differentiates from use case 2 because it includes functionalities to interact with hardware module which are connected to the E-bus of the EtherCAT master or another EtherCAT coupler (i.e: Beckhoff ELK1100), for example, relays, temperature sensors, and servo motor driver, and encoder modules. This use case is the native usage of EtherCAT with hardware of the Beckhoff vendor.

EtherCAT Coupler with Customized Electronic

The use case 4: this scenario incorporates a development of EtherCAT slave that allow to attach customized functionalities implemented in a micro-controller.

EtherCAT Coupler Interfacing with a Sophisticate Subsystem

Use case 5: This is a generic case, in principle, it would be used to adapt future instruments with proprietary protocols to the EtherCAT infrastructure.

IMPLEMENTATION

It was decided to focus our effort and resource to demonstrate the use case 0 and use case 2, because their are the most complicated and more likely scenario for the coming years. The use case 3 poses very low risks and is well demonstrated already in the industry. The use case 4 and use case 5 are variants of the use case 2.

The EtherCAT-CAN coupler will be implemented on top of a XMC4800 micro-controller [6], mainly because the XMC4800 has a MultiCAN controller that could support up to 5 independent CAN buses. Using the SDK of the XMC platform, the ALMA customized CAN protocol is implemented. The choice of the XMC4800 was heavily influenced by past experience of the team, but any micro-controller, FPGA with CAN transceivers could also be a possible alternative.

This design will cover all the different possible scenarios that ALMA may face in the next decade. It allows a coexistence of the different field bus protocols and a smooth transition of the existing architecture to a one that could also support ALMA2030 projects.

As shown in Fig. 3, the testing setup comprises the following equipment: i) a VM with ACS and TwinCAT development environment, ii) a Beckhoff CX2030 IPC EtherCAT master, iii) an Infineon XMC4800 microcontroler EtherCAT slave, iv) an ALMA AMBSI board representing hardware device of ALMA.

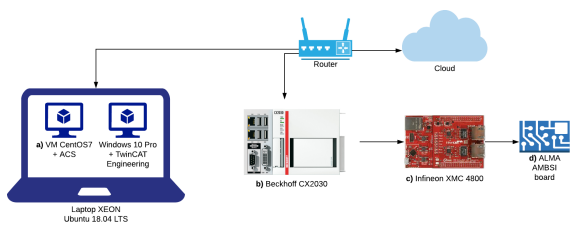


Figure 3: Physical connection of the test bentch.

The objective is to reproduce the existing communication between of an ABM and a hardware devices through the AMBSI interface, but instead sending the customized CAN frame, the data structure will be embedded into a EtherCAT Service Data Object (SDO) frame.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

The SDO protocol in EtherCAT allows to configure (write) and read back values of the Object Dictionary (OD) [7]. It establishes a client-server communication between the EtherCAT master (client) and a EtherCAT slave (server). The server (the XMC-4800 microcontroller) exposes an OD that defines the data structure and the related addresses for the SDO communication. If the amount of the entries of the OD requested by the client exceeds the maximum size, The SDO protocol will segment the data structures in multiples SDO frames. The simplified version of the OD was shown in Fig. 4.

5001:0	ALMA_CAN_Device1	RO	> 6 <
5001:01	AMBSI_SERIAL_NUMBER	RO	0x10e14f7e020800e7
5001:02	AMBIENT_TEMPERATURE	RO	0x10100038 (269484088)
5001:03	PROTOCOL_REV_LEVEL	RO	0x00000000 (0)
5001:04	CAN_ERROR	RO	0x000A0000 (655360)
5001:05	TRANS_NUM	RO	0x0000017F (383)
5001:06	SW_REV_LEVEL	RO	0x00020000 (131072)

Figure 4: Object dictionary.

In order to implement the use case 0 and use case 2, the data flow described in Fig. 5 was implemented, the main components are:

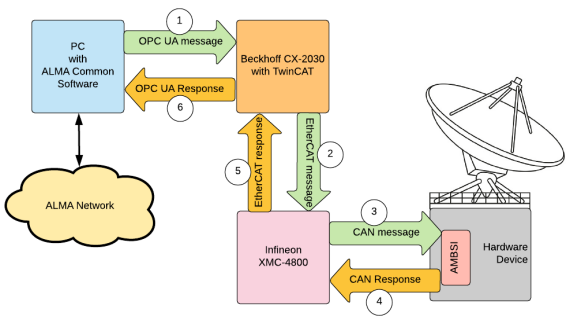


Figure 5: Data flow of the control bus.

OPC UA Device Driver

The existing ALMA software uses Hardware Device Driver approach, in which according to the type of the hardware controller, the corresponding device driver must be developed. From the time being, the most common drivers are the AMBDevice driver and Ethernet Device Driver. For the OPC UA Device driver, the Ethernet Device Driver was extended, and additional OPC UA functionalities were added. As described in Fig. 6 This driver in essentially a OPC UA client that reads/writes data structure defined in OPC nodes hosted by an OPC server. For each entry of the OD, there is a specific node in the OPC server. Within this node, a sub-node is defined for receiving requests from the client (AMBMessage), additional the client subscribes itself to a second sub-node in order to receive the eventual response (AMBResponse).

TwinCAT Application

A TwinCAT application was developed. This application is the EtherCAT master that runs inside of the Beckhoff CX2030. As shown in Fig. 7, the main objective of this application is to serves as a gateway between the OPC

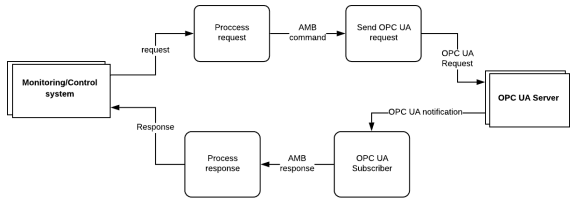


Figure 6: Data flow: ACS to OPC UA section.

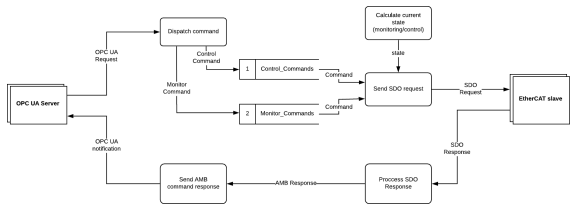


Figure 7: Data flow: TwinCAT to XMC section.

UA interface (external) and the EtherCAT bus (internal). This application implements the OPC UA server with a tree of nodes that represents the OD. Upon an reception of a OPC UA client request, the server queue the corresponding AMBMessage data structure in the control queue or monitor queue accordingly. Two messages processors (see Fig. 7) convert the AMBMessage into SDO data structure and dispatch this message in the EtherCAT bus. In addition, a state machine was implemented in order to separate the dispatch of messages in the queue according to the definition of the TE (see Fig. 1): in each windows of 48 ms, the first 24 ms are reserved for the control requests, and the next 20 ms are reserved for the monitor requests, then 4 ms of idle period follows. In each AMBMessage data structure, the address field maps the sub-index in the OD in a SDO frame. Once the EtherCAT slave (XMC4800) responds the SDO request, the data is extracted and filled inside of an AMBResponse structure which is updated in the corresponding OPC node in the tree of the OPC Server. The node trigger a notification to the OPC client (previously registered to this node) and the data is ready to be retrieved by the OPC client embedded in the OPC UA Device driver.

EtherCAT Slave (XMC4800)

Inside of the micro-controller XMC4800, as shown in Fig. 8, an firmware is implemented to forward each SDO [8] request to the AMBSI card. Upon arrival of an SDO request, the firmware extract the relevant parameters and create an ALMA CAN frame and dispatch it in the CAN bus synchronously, once the respond of the AMBSI card returns, the XMC updates the content in its OD defined its EtherCAT Slace Controller (ESC), and the next EtherCAT frame will take care to bring it back to the EtherCAT master.

RESULTS

The aforementioned implementation allow to generate an end-to-end communication between the logic layer inside of the ACS component and the hardware device located behind the AMBSI board by using COTS hardware and software.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

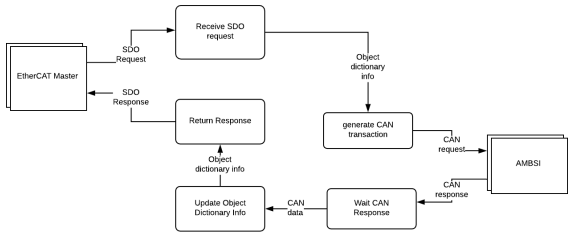


Figure 8: Data flow: XMC to AMBSI section.

The performance tests consist on sending 10000 messages of 6 possible monitor points (see Fig. 4).

As shown in Fig. 9, we built a histogram with the time-of-fly of EtherCAT messages associated with six monitoring points (sub-index 01 to 06) of a hardware device in an antenna. We used the Wireshark software to analyze the TCP packets traffic between the CX2030 and the XMC4800. The packets were processes on a Jupyter notebook to calculate the time-of-fly of each EtherCAT packet. Given the limitations of the sampling time, it was not possible to obtain the EtherCAT propagation times. However, we estimated that it is in the order of nanoseconds, which is negligible for our measurement as the most time consuming in the communication process is related to the CAN communication between the XMC4800 and the AMBSI card. The mean values and the standard deviation of the times of each monitoring point were also calculated and shown in the Fig. 9. The monitor points defined in the OD at the sub-index 1,3, and 6 took 200 µs in average, while the sub-index 2, 4, and 5 took 400 µs in average. The differences between these two groups are because the implementation done inside of the XMC.

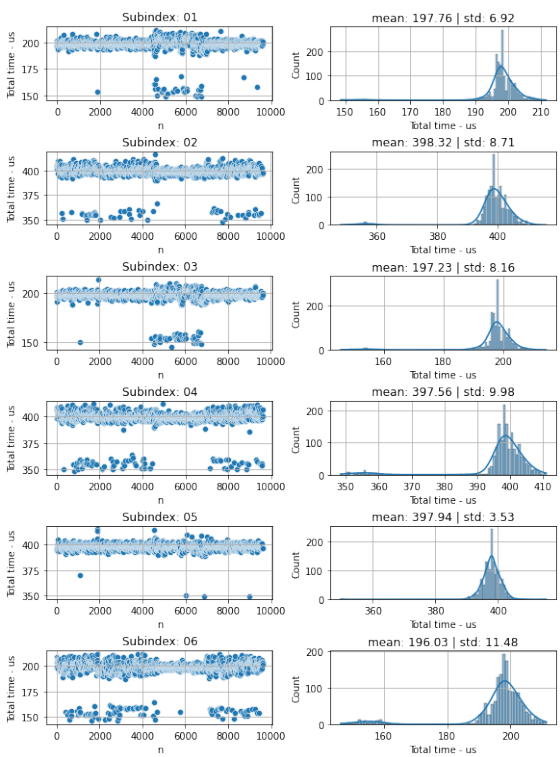


Figure 9: SDO traffic performance.

CONCLUSION AND FUTURE WORKS

The proposed design demonstrated to be an viable alternative to replace the existing control bus in ALMA. Based on a open standard such as EtherCAT, it offers a ready to use framework to design the control interface of new hardware of the ALMA2030 projects, these hardware will behave as EtherCAT slaves. During the transition period, it was expected to have a combination of existing CAN based hardware device and EtherCAT based device in operation. This scenario can be supported by the design proposed in this paper. The feature of having an OPC UA interface between the Hardware Device Driver and the EherCAT master (replacement of ABM) allows more efficient usage of the technical time between software engineers and hardware engineers in the observatory. The hardware engineers don't need to use the entire ALMA software for their specific hardware troubleshooting activities, which currently block the software testing activities of the software engineers in the observatory. Finally, the ALMA ICD definition fits perfectly in the concept of the OD dictated by the EtherCAT CoE, therefore minimal impact is expected in the Hardware Device layer.

Nevertheless, the SDO structure suits to the purposes of Monitoring and Control in the ALMA Control bus, we believe more performance can be achieved by using the PDO structure. The SDO can be reserved for setup commands, and low data rate monitoring points, such as temperature sensors. But the PDO structure can be reserved for high data rate monitoring points, such as, vibration sensors, encoders values, etc. The concept of using PDO is under analysis. Additionally, the TE infrastructure also needs to be improved in this design as same as the possibility to use the MultiCAN infrastructure of the XMC4800. Currently, only one of the possible six channels are being exploited.

REFERENCES

[1] EtherCAT Technolog Group, https://www.ethercat.org/default.htm

[2] Proyecto QUIMAL190009, https://cemcc.ufro.cl/quimal/

[3] G. Chiozzi, B. Jeram, H. Sommer, *et al.*, "The ALMA common software: a developer-friendly CORBA-based framework", in *Proc. SPIE*, vol. 5496, pp. 205–218, 2004. doi:10.1117/12.551943

[4] "CAN Bus Specification 2.0", Robert Bosch GmbH, 1991.

[5] R. Amestica, B. Gustafsson, and R. Marson, "Time synchronization within the ALMA software infrastructure", in *Proc. SPIE*, vol. 6274, pp. 338–346, 2006. doi:10.1117/12.670298

[6] XMC4800 32 bit Arm Cortex Microcontroller, https://www.infineon.com

[7] "EtherCAT Modular Device Profile", ETG.5001.6220 S (D) V1.0.5. https://www.ethercat.org/download/documents/ETG5001_6220_V1i0i5_S_R_IO-LinkMaster.pdf

[8] "EtherCAT Master Classes", ETG.1500 D (R) 1.0.2. https://www.ethercat.org/download/documents/ETG1500_V1i0i2_D_R_MasterClasses.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

THE STATE OF CONTAINERIZATION IN CERN ACCELERATOR CONTROLS

R. Voirin*, T. Oulevey, M. Vanden Eynden, CERN, Geneva, Switzerland

Abstract

In industry, containers have dramatically changed the way system administrators deploy and manage applications. Developers are gradually switching from delivering monolithic applications to microservices. Using containerization solutions provides many advantages, such as: applications running in an isolated manner, decoupled from the operating system and its libraries; run-time dependencies, including access to persistent storage, are clearly declared. However, introducing these new techniques requires significant modifications of existing computing infrastructure as well as a cultural change. This contribution will explore practical use cases for containers and container orchestration within the CERN Accelerator Controls domain. We will explore challenges that have been arising in this field for the past two years and technical choices that we have made to tackle them. We will also outline the foreseen future developments.

CONTAINERS IN CONTROLS SYSTEMS

Containers in a Nutshell

Namespaces started being implemented in the Linux kernel in the 2000s. They provide isolation features on multiple levels: while the mount namespace prevents the process from accessing the rest of the Linux filesystem, the Process ID (PID) namespace creates an independent PID number space where the isolated process is given PID 1. There are eight namespaces in total: mount, PID, network, interprocess communication, time, time-sharing, user and cgroup.

Containers can be seen as industrialization of these isolation mechanisms, where the use of namespaces is concentrated in a single "containerization layer". A containerized application runs in an isolated manner, and requires its dependencies (libraries) to be embedded (Fig. 1).

The Open Container Initiative (OCI) provides three specifications defining how containerized applications are stored (the Image Format Specification), run (the Runtime Specification) and distributed (the Distribution Specification) [1].

CERN Use Cases

Having been used in industry for some time already, it was clear that the benefits brought by containers could translate to CERN's Accelerator Control system. In April 2020, a project was launched to introduce containers to the Accelerator Controls landscape, in order to bring added value in a variety of areas.

Firstly, is the ability to decouple from the underlying host operating system and the flexibility this brings. At CERN,

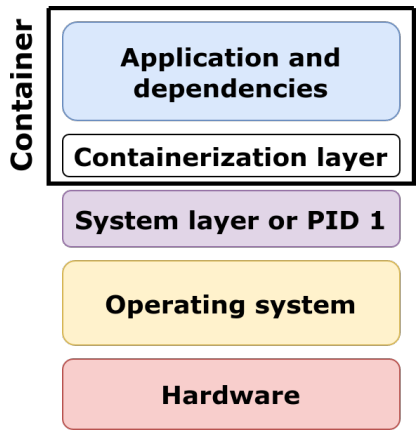


Figure 1: Overview of a containerized software stack.

WinCC OA is used to manage many industrial SCADA systems [2]. Version 3.16 officially runs on CentOS 7 and Red Hat Enterprise Linux (RHEL) 7, while 3.18 will be made to run on Red Hat Enterprise Linux 8. Due to the massive number of WinCC OA applications, many of them for critical systems, migration from version 3.16 to 3.18 must be applied progressively. In contrast, the operating system upgrade from CentOS 7 to the next platform, will concern all hosts at once. For this case, deploying WinCC OA in containers will make it possible to run version 3.16 in a CentOS 7-based container, while the underlying host is already upgraded to RHEL 8 or a derivative.

Containerization is also becoming a *de facto* standard for companies to deliver software to their clients. For example, SourceGraph [3] is used in the CERN Controls software community to index code, quickly search through it, and create statistics. All deployment options for self-hosted instances of SourceGraph are container-based.

Other software are delivered in container images for simpler deployment and upgrades. This is the case for the Nexus Repository Manager, which is used at CERN to manage Python libraries [4].

Another advantage of container-based software delivery is the idempotent behaviour of a product between development and operational environments. This can translate into two ways: streamlining the creation of development and operational releases in a similar way, and easily providing a containerized test setup. For the LHC Orbit Feedback (OFB), the latter has proven cost and time effective. Instead of running many test instances on a dedicated server and configuring them via a database, it is possible to run a local containerized copy and feed it directly with the desired parameters.

Containers are also a way to distribute software that can easily run regardless of the Linux distribution, or even the

* remi.voirin@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

operating system, as the Docker company created a desktop-oriented product for MacOS and Windows [5]. This has been successfully used for multiple PyQt training courses at CERN. It allows participants to run a particular Integrated Development Environment (IDE), faster, and with a lighter memory footprint than with a virtual machine.

Finally, there are many use cases where a clean and predictable software environment needs to be quickly generated, for instance in the context of Continuous Integration (CI) and Continuous Deployment (CD). A containerized environment is the most effective way to get this done.

Issues and Limitations

The differences between a virtual machine and an OCI container can be summarized in the most simple terms by the fact that running a virtual machine implies executing a guest operating system (kernel). Nevertheless, even in the absence of a kernel, container images need to provide shared libraries and basic building blocks like libc in order for an application to run properly. These components are provided by a Linux distribution, which means that they need to be managed and kept up to date.

While most containers will run the CERN supported Linux distribution, external containerized applications are typically made with others like Alpine and Debian. Importing such containers is similar to adding computers running these alternative distributions in the Controls environment. It therefore implies a need to track and address security issues for a broader set of software.

Another issue is the growing complexity of the overall software stack. In case of unexplained latency, lost packets, or unexpected behavior, debugging involves more components to analyze. For instance, using containers implies having virtual interfaces to forward packets to them. Capping CPU and memory usage involves the use of cgroups, which need to be understood and mastered in the system administration team. Using additional debugging techniques may be required, like eBPF [6].

Containerization is efficient for scaling and reproducing software instances. However, when using proprietary software, careful attention needs to be given to ensure licensing terms are not violated.

Required Components

In order to provide functional containerization in a Controls system, three components are required:

- a set of base images which will be used by developers as a foundation to package and deploy containerized software;
- a container registry to store images;
- a container engine, which is the piece of software made to run containers on hosts.

BASE IMAGES

Base images are made to replicate a Controls host (technical console or server) running CERN CentOS 7 in a lighter fashion. The "acc_cc7" base image is made from scratch by setting up a few packages (e.g. the YUM or DNF package manager, a text editor) in a directory, then loading the directory into the OCI image format.

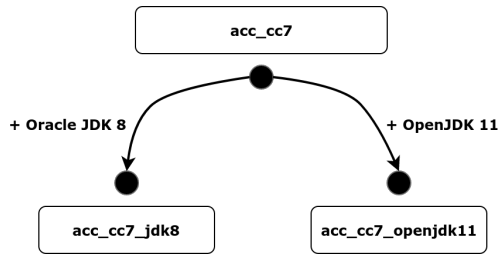


Figure 2: CERN CentOS 7 base images.

CERN Accelerator Controls images are configured to satisfy specific requirements, such as all Controls hosts running software from a set point in time called the "snapshot date" to give a homogeneous experience. This concept is applied within the container images thanks to a custom YUM repository configuration.

Images containing Java Development Kits (JDKs) are built on top of the acc_cc7 base image in order to run Java-based software, either with the Oracle JDK 8 or with OpenJDK 11 (Fig. 2). This is done by running an Ansible playbook which sets up the relevant Java packages and their configuration files within a temporary container, then extracting the resulting directory structure and placing it in a clean container image.

The overall process is launched every day in a CI/CD pipeline. It is light enough such that when a new major version of RHEL/CentOS is available, minimum changes are required to update the image generation process.

CONTAINER REGISTRY

Though the container registry can be described as simply as a repository of images, there are specific requirements that need to be satisfied. It must be able to store internal images (i.e. base images and containerized applications built on top), as well as images which come from public registries like the Docker Hub. Such images can not be pulled directly from Controls hosts, as they have no access to the Internet, hence the registry acts as curated proxy in this sense.

For security reasons, pushing images to the registry should be restricted to a specific process, while pulling should be done anonymously, allowing easy integration in automated software deployment processes. Following a market survey and discussing with other CERN teams working in this domain, it was decided to use the Harbor registry [7] (Fig. 3).

Harbor deployment is managed by the cloud infrastructure team within the IT Department. It fulfills the main functional requirements and it integrates well with additional security features such as an image validation policy and vulnerability scanning.

Validation Policy and Vulnerability Scanning

The Controls computing infrastructure team has to find a balance between delivering a smooth experience to develop-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

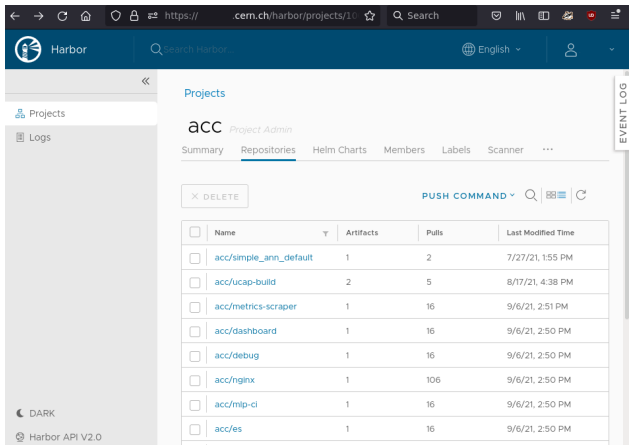


Figure 3: The Harbor registry.

ers, and at the same time, enforcing a security policy where only a certain set of images are allowed.

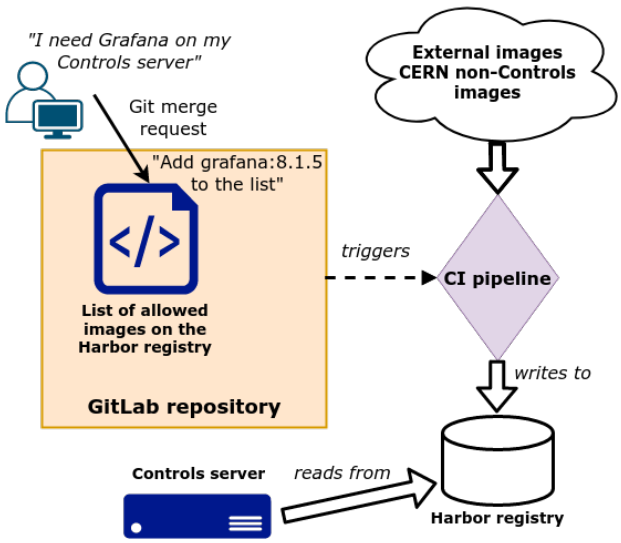


Figure 4: Image request and validation mechanism.

The Controls computing infrastructure team has designed a mechanism where developers can call a script which, after filling appropriate parameters, will create a merge request on a GitLab repository to add a new image to the Harbor-based Controls container registry (Fig. 4).

Only images that are relevant for operation are accepted. Later in the chain, vulnerability scanning is invoked from the Harbor registry, giving insight on the security status of the image. In 2021, Trivy [8] is the tool used to perform these scans (Fig. 5).

CONTAINER ENGINE

When the use of containerization technologies exploded in the 2010s, Docker was the default choice for a container engine. Since then, the company behind Docker has adapted its business model, whereby the use of some products need to be paid for [9]. This change in policy, combined with the

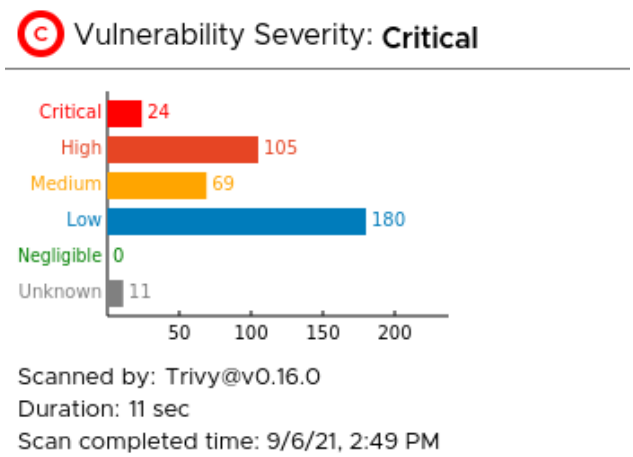


Figure 5: Scan result of a critically vulnerable image.

potential for a sudden price increase, raises concerns when it comes to choosing a container engine for Controls, that must remain stable over the course of a full LHC Run (e.g. 5 years).

Docker revolves around a UNIX daemon running as root. It creates an additional attack surface on every Controls host where it would be deployed, and also consumes some RAM even when no container is launched. It becomes a single point of failure where containers are children of this process, and in case of any issues, they become orphans (PID 1 becomes the parent).

In the meantime, OCI alternatives have emerged, like Podman [10]. This software is supported by Red Hat and is included in the RHEL distribution and its derivatives.

Podman does not require a daemon, which addresses the problems stated above. Being daemonless allows easy integration with LUMENS, the Controls tool to manage and run user processes. With only slight adjustments, it is possible to treat container startup similarly to any other systemd service.

Podman can be configured to run rootless containers, which means that they run completely within the scope of a UNIX account. This provides an increased level of security and practicality, as root may not be involved at all.

Rootless Containers

Even if all the required components are present (i.e. a recent enough Linux kernel supporting namespaces and the container engine to use them), some configuration is still required to enable rootless containers. First, user namespaces need be enabled. This is done thanks to the the max_user_namespaces kernel parameter, which is set to 0 by default. A high enough value needs to be set to accommodate as many containers as a Controls host can support. At CERN, this is set to 8192 giving room to run a few hundred containers.

During the daily Ansible run, which enforces the general configuration on Controls hosts, the list of users who can run rootless containers will be extracted from LDAP. This list is then written in the /etc/subuid and /etc/subgid

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

files in the proper format: <username>:<start of subuid range>:<number of subuids>. To avoid collisions, subuids and subgids are calculated by shifting the user ID 8 bits to the left, and leaving the last 11 bits for subuids and subgids (Table 1). Therefore, the number of subuids and subgids is constant for all users: $2^{11} - 1 = 2047$.

Table 1: Example of subuid range for a given UID

User ID	Beginning of range	End of range
1000	1000 << 11 = 2048000	2050047

Finally, user-writable directories in /opt are created to store images, containers and their metadata. This is done by again taking the list of relevant users in LDAP, then making sure these directories exist for each one of them with proper Ansible tasks.

CONTAINER ORCHESTRATION AND THE FUTURE OF THE ACCELERATOR DATA CENTER

Moving Away from Bare Metal

The operation of the CERN Accelerator complex relies on a dedicated data center that was put in operation for the LHC start-up in 2008. A total of about 400 high-availability servers are used for the operation of critical LHC infrastructure systems such as Cryogenics as well as for all systems required for beam operation.

Over the years, this critical infrastructure remained as a 100% bare metal facility. Consequently, each process required for the operation of the CERN Accelerator complex runs on a dedicated server with a fixed IP address. So far this infrastructure offered excellent availability figures, however, it is clear that it can be optimized to make better usage of the overall computing power and to be more agile in terms of hardware maintenance effort (the current annual system administration effort takes up to 4 days). This is where virtualization and container orchestration technologies enter the game.

Relevance of Container Orchestration in Controls

A simple way to summarize container orchestration is that instead of abstracting a single computer, an entire computing infrastructure is abstracted. An orchestrator takes care of the placement, scheduling, scaling, failover and health monitoring of containers. It also provides an API to interact with it, and accepts input with declarative configuration (generally in YAML format).

Stateless web services are one of the best use cases for orchestrated environments as they benefit from horizontal scalability, redundancy, and load balancing techniques without manual configuration of additional software like HAProxy and nginx.

Control systems are usually made of very diverse hardware and software with variety of monitoring and software deployment methods. Since it is not feasible for Controls infrastructure to be entirely deployed in orchestrated containers, it means that multiple deployment and mechanisms must be maintained, resulting in a growing stack and overall, a more diverse and complex infrastructure to manage.

Another point of concern is related to security: lots of institutions prevent their Control systems from directly accessing the Internet. It means that the orchestrator has to be physically deployed on-site, and managed by a local team. While deploying software on orchestrators is convenient and eases developers' lives, the underlying infrastructure is quite complex. It implies dedicating additional time to maintain it, increasing total costs, unless orchestrated containers completely replace the legacy stack.

Finally, one of the best selling points for container orchestration in industry is complete scalability. When a company relies on microservices that need to be replicated to manage sudden load increases, and for which the computing power can be rented from one of the big cloud providers, it is easy to understand the interest in this technology. Benefits are less apparent when servers have to be physically provisioned on site (i.e. they have a fixed cost), and that computing resource don't need to be regularly redistributed to different applications over time.

Previous Studies

There were two previous attempts at evaluating container orchestration for CERN Accelerator Controls in the late 2010s.

In 2018, the LHC Operations software team attempted to set up Kubernetes clusters to measure the performance of the LHC Coupling Analysis Service in containers [11]. Despite the use of infrastructure as code with Ansible, the complexity of deploying Kubernetes clusters in an environment without Internet connectivity was clearly highlighted.

Between 2018 and 2019, a study took place in the Accelerator Controls Group to evaluate container orchestration in a specific context, including the use of Nexus as an image registry, and creation of a proof-of-concept of base image. The conclusion, was that in-house deployment of Kubernetes was dismissed, due to human resources considerations and incompatibility with the accelerator schedule. Major Kubernetes versions were released every three months, whereas some Control system upgrades cannot be scheduled to take place more often than once per year. Upgrading Kubernetes on an annual basis could be quite risky, representing giant leaps across multiple versions.

CERN IT Container Orchestration Infrastructure

Outside of Controls, OpenStack is used as the backbone of the CERN IT data center since 2012. The IT container deployment platform is based on OpenStack technology, together with an open-source component called Magnum [12]. Magnum is an Openstack API service developed by the OpenStack Container Team, with numerous patches from CERN contributors. It allows to run orchestrators such as Docker Swarm or Kubernetes. This section will give more

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

details on the Kubernetes use case, which is the main orchestrator in use. Nevertheless, it is important to note that Magnum templates can be added for different orchestrators, without changing the underlying architecture and hardware.

CERN OpenStack [13] offers a self-service solution to get an orchestrator running in the private cloud. CERN users can request a cluster with various storage, compute (CPU, GPU) or traffic routing options. On the storage front, CERN offers support for CephFS [14], CVMFS [15] and EOS [16], which are quite different from industry standards.

Magnum can spawn a cluster on virtual machines or the bare metal infrastructure thanks to the integration with other OpenStack APIs. Once created, CERN users have full control on their clusters and can use industry standard tools to access the Kubernetes APIs and deploy their applications. Around 650 clusters are running at CERN [17], using around 13,000 cores, 30 TB of RAM and more than 150 PB of local storage. Under the hood, the operating system is Fedora CoreOS, which is an automatically-updating, minimal operating system for running containerized workloads securely and at scale.

Helm [18] is a package manager for Kubernetes and gives users the ability to deploy existing sets of containers with almost no code or knowledge of the underlying infrastructure. Users are encouraged to store and deploy their applications using the CERN internal repository.

For monitoring, the IT Department largely relies on Prometheus [19] for gathering cluster metrics, which is provided by default. Users can also add their own custom metrics.

This paper wouldn't be complete without a mention of the HashiCorp product, Nomad [20]. Nomad is a modern, easy to install and lightweight workload scheduler. By creating a pool of compute, storage, and networking, Nomad can decide where it's most efficient to run tasks. The deployment consists of a single binary and the built-in task drivers plug-in allows to run all sorts of workloads to also bring orchestration benefits to existing services. A few teams in the IT Department have put in place self-managed solutions based on the open source version of Nomad.

As shown in this section, container orchestration is a complex subject and requires interactions with a lot of existing IT infrastructure components. The investment to have something working in an environment that spans servers, networks, storage and development processes is very challenging for small teams.

Container Orchestration and Accelerator Data Center Management

In 2021, service continuity and recovery is considered as a high-priority topic with the CERN accelerator domain. One of the means to achieve this from the computing infrastructure perspective, is to allow critical services to easily recover, by starting them from a different data center.

Accelerator Controls back-end services are provided by around 400 servers in a single location. IP networks are physically segmented in the data center, no anycast range is available, and enabling routing protocols up to the host isn't available. Storage is also provided by NFS servers that are single points of failure.

Prior to introducing technical solutions for computing service continuity and flexible data center management, it will be required to invest in an evolution of the network infrastructure, as well as deploying scalable storage.

The hardware team has identified: key servers to physically duplicate, required rack space, power consumption, and expected recovery times from daily backups. It was also crucial to focus on logical aspects and how services and data can be transferred, or kept in a working state, when moving from one data center to another.

Virtualization and container orchestration were analyzed (Fig. 6). To sum up, all these technologies solve the same problem which is, providing service continuity in the case of a catastrophic event in a data center. They also enable the hardware team to manage servers in a more generic way, by looking at a global set of systems instead of individual treatment for each of them. Thanks to virtualization and container orchestration, physical computing resources can also be shared, using a logical split for different use cases. Currently, a team requiring back-end resources needs to ask for a full server.

While orchestrators provide APIs to deploy applications and offer features like load-balancing and autoscaling, they can only be used to host cloud-ready applications. Currently, only a few use cases of specific Controls sub-systems are ready e.g. the Machine Learning Platform (MLP) [21] and the Unified Controls Acquisition and Processing (UCAP). Stateless web services could be easily transferred as well. However, many Controls applications would need significant architectural changes. Also, one third of Accelerator data center servers are dedicated to running WinCC OA backends to control SCADA systems, and this specific software won't be ready for orchestration for some years to come.

From the development and deployment perspective, CI/CD integration is better with orchestrators, but they imply rethinking process deployment and monitoring, currently done using LUMENS. CI/CD integration would be the same between current servers and potential virtual machines.

Kubernetes, as provided by the CERN IT Department, runs as a proof-of-concept. It would need further refinements in order to run mission critical applications, such as the ability to run multiple cluster masters to mitigate the impact of localized hardware failure. Currently, if one server fails, orchestrated applications still run but can't be easily stopped or restarted. Other orchestrators like OpenShift and Nomad can be deployed at an additional cost, as training / consultancy for the system administration team would be needed in order to deploy an optimal setup.

Container orchestration will not avoid the fate of non-cloud-ready applications in the case of a disaster in the CERN Accelerator data center. Given the large number of use cases that cannot be orchestrated in the foreseeable future, flexibility should come with the use of virtual ma-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

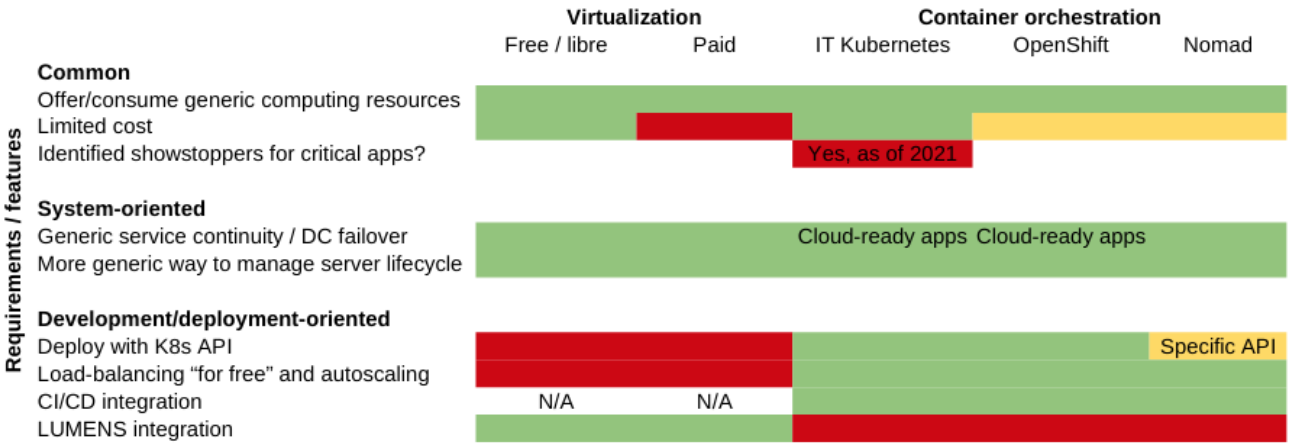


Figure 6: Summary of options for computing service continuity and flexible data center management.

chines, for which floating IPs will allow services to quickly continue in a second data center.

CONCLUSION

Containerization applies to different concepts and can be used on two different scales: plain containerization (by deploying on the current hardware and system stack), and container orchestration (by generically dedicating computing power to it).

At CERN, plain containerization is in place with fully functional base images, a container registry to store images, as well as a container engine configured to run rootless containers. It is used for operation and is fully supported.

Orchestration is deployed as a proof-of-concept, with software infrastructure provided by the IT Department, but it is currently not ready to run operational services. Introducing container orchestration on a massive scale implies a full review of software development practices. It can be used as a way to manage a data center in a more flexible manner, provided that the necessary techniques reach a critical mass adoption.

ACKNOWLEDGEMENTS

The author expresses his sincere appreciation to Karim Ben Othman for helping set up containerization blocks and writing effective Python libraries.

The author thanks Frank Locci for his work on integrating Podman containers in LUMENS, as well as Steve Traylen for providing a solution for management of subuids and subgids.

REFERENCES

[1] Open Container Initiative GitHub repository, https://github.com/opencontainers/.

[2] P. Golonka, L. Davoine, M. Zimny, and L. Zwalinski, "Virtualisation and Software Appliances as Means for Deployment of SCADA in Isolated Systems", presented at the 18th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS'21), Shanghai, China, 2021, paper THPV049, this conference.

[3] Install Sourcegraph, https://docs.sourcegraph.com/admin/install

[4] Nexus3 container image, https://hub.docker.com/r/sonatype/nexus3/.

[5] Docker Desktop, https://www.docker.com/products/docker-desktop

[6] eBPF, https://ebpf.io/.

[7] Harbor, https://goharbor.io/.

[8] Trivy GitHub repository, https://github.com/aquasecurity/trivy

[9] Docker Subscriptions, https://www.docker.com/blog/updating-product-subscriptions/.

[10] Podman, https://podman.io/

[11] Marc Benedi and Andrea Calia, "Construction of a Kubernetes cluster in the CERN Technical Network and automating its deployment with Ansible", 2018. https://cds.cern.ch/record/2636772

[12] CERN Openstack Magnum git repository, https://gitlab.cern.ch/cloud-infrastructure/openstack-magnum

[13] CERN Openstack users documentation https://clouddocs.web.cern.ch/containers/README.html

[14] Container Storage and CephFS: CSI CephFS, https://techblog.web.cern.ch/techblog/post/container-storage-cephfs-csi-part1/.

[15] CVMFS CSI driver, https://gitlab.cern.ch/cloud-infrastructure/cvmfs-csi

[16] EOS storage provisioner, https://clouddocs.web.cern.ch/containers/tutorials/eos.html

[17] Statistics on Kubernetes clusters running in the CERN infrastructure, https://monit-grafana-open.cern.ch/d/ti8EaxwZk/kubernetes-public?orgId=16

[18] The package manager for Kubernetes, https://helm.sh

[19] Prometheus is an open-source systems monitoring and alerting, https://prometheus.io/.

[20] Nomad by HashiCorp, https://www.nomadproject.io/.

[21] J.-B. de Martel, R. Gorbonosov, and Dr. N. Madysa, "Machine Learning Platform: Deploying and Managing Models in the CERN Control System", 18th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS'21), Shanghai, China, 2021, paper MOBL03, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL04

KUBERNETES FOR EPICS IOCs

G. Knap, T. Cobb, Y, Moazzam, U. Pedersen, C. Reynolds
Diamond Light Source, Oxfordshire, UK

Abstract

EPICS [1] IOCs at Diamond Light Source (DLS) [2] are built, deployed, and managed by a set of in-house tools that were implemented 15 years ago. This paper will detail a proof of concept to demonstrate replacing these legacy tools and processes with modern industry standards.

IOCs are packaged in containers with their unique dependencies included.

Container orchestration for all beamlines in the facility is provided through a central Kubernetes cluster. The cluster has remote nodes dedicated to each beamline that host IOCs on the beamline networks.

All source, images and individual IOC configurations are held in repositories. Build and deployment to the production registries is handled by continuous integration.

Development containers provide a portable development environment for maintaining and testing IOC code.

INTRODUCTION

The approach presented here has 5 main themes:

1. **Containers**: package each IOC with its dependencies and execute it in a lightweight virtual environment. [3]
2. **Kubernetes**: centrally orchestrates all IOCs at the facility [4].
3. **Helm Charts**: deploy IOCs into Kubernetes and provide version management [5].
4. **Repositories**: Source, container and Helm repositories hold all of the assets required to define a beamline's IOCs.
5. **Continuous Integration**: source repositories automatically build containers, Helm charts and deliver them to package repositories.

An initial proof of concept (POC) has been implemented at DLS on the test beamline BL45P. All the source code for the proof of concept, plus documentation and tutorials can be found in the GitHub organization epics-containers [6].

SCOPE

The POC initially targets Linux IOCs. This includes IOCs that communicate with their associated devices over the network, as well as those that connect to local devices through USB, PCIe etc. It does not include provision for Operator Interfaces (OPIs) as these vary greatly between facilities. Future plans include:

1. Support OPIs by having a 2nd container for each IOC instance that serves OPI files over HTTP.
2. Supporting RTEMS hard IOCs: using a containerised developer environment shared with soft IOCs.
3. Support Windows IOC development through a similar approach to RTEMS.

CONTAINERS

A class of IOCs that connect to a particular class of device will all share identical binaries and library dependencies; they will differ only in their start-up script and EPICS database. Thus containerized IOCs may be represented as follows:

1. **Generic IOC**: A container image for all IOCs that will connect to a class of device.
2. **IOC Instance**: a Generic IOC image plus unique instance configuration. Typically the configuration is a single start-up script only.

This approach means that the number of container images is kept reasonably low and they are easier to manage.

Image Layering

Container images are typically built by layering on top of existing images.

For the POC, an image hierarchy is used to improve maintainability as shown in Fig. 1 below.

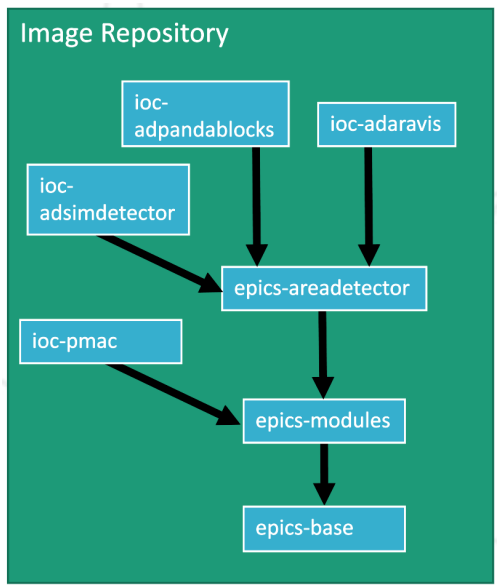


Figure 1: Image hierarchy for the generic IOCs in the current proof of concept.

EPICS base [7] and essential tools are compiled inside one image; the most commonly used support modules (primarily Asyn [8]) and the AreaDetector [9] framework also have their own images. Generic IOC images are then

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

leaves in the hierarchy and are based upon the appropriate dependencies.

Images also have internal layering and every layer is shared between all instances of IOCs, both in image repositories and at runtime. Fig. 2 shows and example of this internal layering.

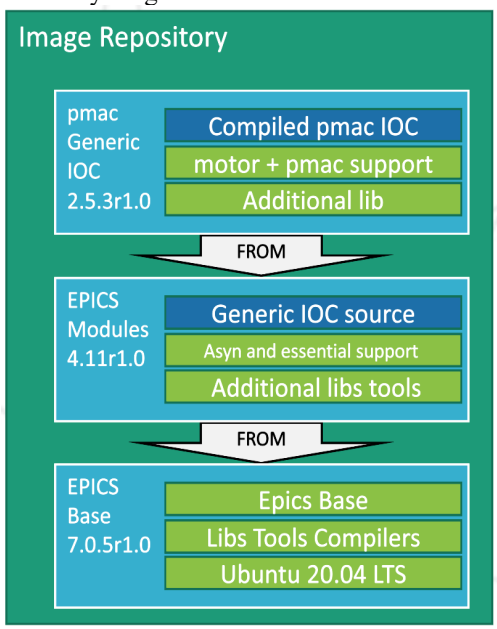


Figure 2: Example of the layered nature of a container image for the pmac motor controller generic IOC.

Developer Container

To save on resources at runtime, all images are built using a staged Dockerfile [10]. There are two targets: a developer target which includes all of the compilers and tools used to build the runtime assets; and a runtime target containing the minimum assets to run the generic IOC.

The developer targets are used to provide an environment for developers to compile and debug the IOC and its dependent support modules. This provides a portable development environment and means there is no need to replicate the tool chain directly on developer workstations.

KUBERNETES

A central Kubernetes cluster is used to orchestrate the IOC instances for all beamlines in the facility. It provides the following functionality that is typically handled by separate tools:

- Auto start IOCs when servers come up
- Manually Start and Stop IOCs
- Monitor IOC status and versions
- Deploy versions of IOCs to the beamline
- Roll back to a previous IOC version
- Allocate the server which runs an IOC
- View the current log
- View historical logs (via graylog at DLS)
- Connect to an IOC and interact with its shell
- Debug an IOC (by starting a bash shell inside its container)

Cluster Topology

A single multi-tenant central cluster runs all of the worker nodes. This provides centralized management of all beamlines and other services. The High Availability (HA) control plane has 3 virtual servers distributed across 3 physical servers.

Each beamline has its own physical servers that are configured as remote worker nodes to the central cluster. This means that:

1. IOC instances are close to the hardware that they communicate with, avoiding network bottlenecks. This is important for high bandwidth IOCs such as area detectors.
2. IOC instances may be given affinity to a specific server and communicate directly with hardware connected to that server (e.g. a USB device)
3. IOC instances reside on the same subnet as the beamline's Channel Access (CA) clients and any network attached devices. This is a requirement for CA and some network attached devices (see below).

Each beamline has its own Kubernetes namespace and user id in which all the IOC instances will run. This provides isolation between the beamlines.

Container Networking

Containers use namespaces to isolate their use of system resources. This is an important feature for building reliable, scalable and secure applications. However, EPICS IOCs rely on network protocols that may not suit network isolation because they do not easily pass through Network Address Translation (NAT). For this reason the POC foregoes virtual networks and uses the native networking of the host server.

Channel Access (CA) and pvAccess (PVA) are the primary protocols for communication between IOCs and clients. Both protocols require a broadcast in order to establish initial communication. The broadcast does not work via NAT to a virtual LAN.

Other protocols between IOCs and devices had issues with virtual networks. e.g. GigE Vision Stream Protocol (GVSP) establishes a connection by passing an IP address and port number in the application layer and therefore does not pass through a NAT.

Workarounds to the protocol issues were investigated on a case by case basis but it became clear that the only reliable solution was to use native networking within a single subnet. This is a slight concession to security, but is no worse than traditional IOC deployment. All other namespaces are applied to IOC containers and they are isolated from the host in all respects except network.

HELM

The POC supplies a Helm Chart Library that describes all of the Kubernetes resources required to deploy an IOC instance to a beamline. Each Beamline has a source repository that specifies a Helm Chart for each of its IOC instances. The beamline source need only refer to the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL04

Library and supply a few parameters to define the unique properties of the IOC instance – most notably its start-up script. The Library has templates for the following resources:

1. A Deployment [11] which makes sure 1 instance of the IOC Pod is always running.
2. The Pod [12] is described within the Deployment YAML. It includes a reference to the Generic IOC image it uses to launch the container.
3. A ConfigMap [13] which is mounted as a folder containing the unique configuration for the IOC instance. This typically contains only a start-up script.
4. A Persistent Volume Claim [14] which is mounted to provide persistent storage across IOC restarts and upgrades. This is used to hold autosave data.
5. Note that there is no Kubernetes Service [15] associated with the IOC since it uses native networking and is addressed via the host's IP address directly.

Helm command line functions are used to deploy IOC instances to a cluster and manage multiple versions of IOCs within the cluster.

Helm charts may be stored in a registry. For the POC the registry is provided by the epics-containers GitHub organization. New versions of IOC Helm charts are released to the registry and then deployed from it. Fig. 3 demonstrates this release process.

REPOSITORIES

All of the assets required to manage a set of IOCs for a beamline are held in repositories. Thus all version control is done via these repositories and no shared file-systems are required. The classes of repository are as follows:

1. **Beamline Source**: (1 per beamline) holds the source for Helm charts for each IOC instance. Also hosts the continuous integration (CI) steps to generate Helm charts and publish them to the Helm repository.
2. **Container Image Source**: (1 per Generic IOC) holds the Dockerfile that describes the contents of a Generic IOC. Also hosts the CI to generate an image from the Dockerfile and publish to the image repository.
3. **Helm Repository**: (1 per IOC Instance) holds the published IOC Instance Helm Charts ready for deployment to Kubernetes.
4. **Image Repository**: (1 per Generic IOC) holds the Generic IOC container images and their dependencies.

CONTINUOUS INTEGRATION

All published assets in this process are generated via CI (see Repositories above). Every published asset has its own version number and its own source repository. When releasing an asset the developer will version tag the source repository. This causes the CI to build the source, publish the result and tag it with the same version number.

The POC uses GitHub Actions [16] to implement its CI and publishes assets to GitHub Packages [17]. However, during development GitLab CI [18] and Google Container Registry [19] were also tested.

CONCLUSION

The POC demonstrates that it is possible to deploy and manage IOCs for a beamline using only standard open source tools such as Kubernetes, Helm and GitHub. The test beamline BL45P has successfully deployed its IOCs using this approach and required no custom software or scripts to do so.

DLS will continue to develop this approach as a possible site wide solution. The epics-containers GitHub organization will be continuously updated to track progress. The organization is also available for contribution from EPICS developers at other sites.

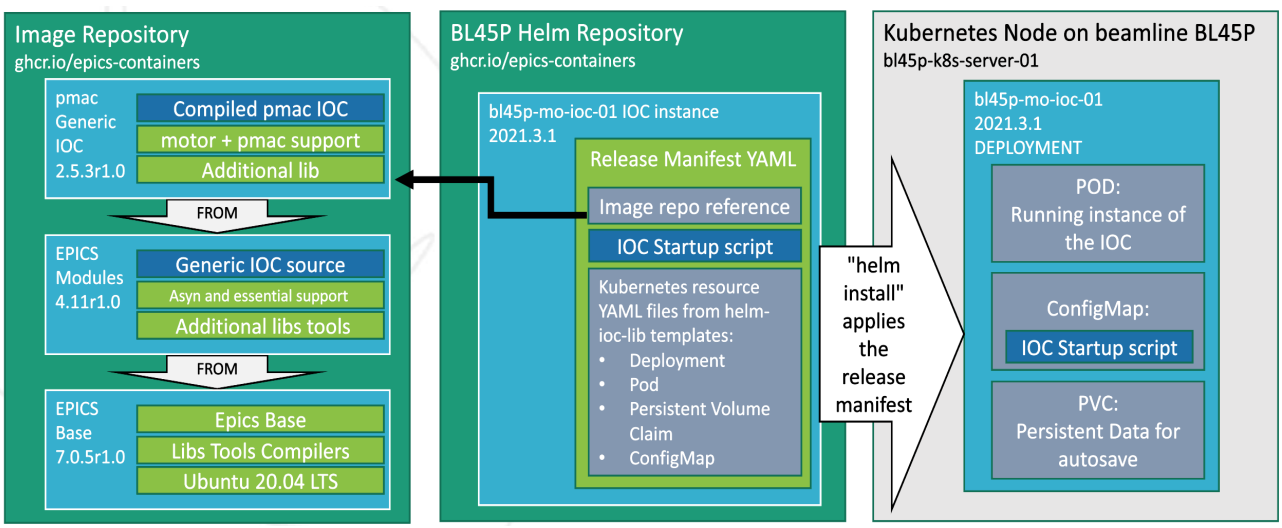


Figure 3: Example of a motion IOC (bl45p-mo-ioc-01) showing how the generic IOC image is built up, packaged with deployment details into a Helm chart, and then deployed to a Kubernetes node.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] EPICS, https://epics-controls.org/

[2] R. P. Walker, "Commissioning and Status of the Diamond Storage Ring", in *Proc. 4th Asian Particle Accelerator Conf. (APAC'07)*, Indore, India, Jan.-Feb. 2007, paper TUYMA03, pp. 66-70.

[3] Open Container Initiative, https://opencontainers.org/

[4] Kubernetes, https://kubernetes.io/

[5] Helm Charts, https://helm.sh/docs/topics/charts/

[6] EPICS Containers, https://epics-containers.github.io

[7] EPICS Base, https://epics-controls.org/resources-and-support-/base/

[8] EPICS Asyn, https://epics-controls.org/resources-and-support/documents/howto-documents/device-support-asyn-driver/

[9] EPICS areaDetector, http://cars9.uchicago.edu/software/epics/areaDetector.html.

[10] Staged Docker Builds, https://docs.docker.com/develop/develop-images/multistage-build/

[11] Kubernetes Deployment, https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

[12] Kubernetes Pod, https://kubernetes.io/docs/concepts/workloads/pods/

[13] Kubernetes Config Map, https://kubernetes.io/docs/concepts/configuration/configmap

[14] Kubernetes PVC, https://kubernetes.io/docs/concepts/storage/persistent-volumes/

[15] Kubernetes Service, https://kubernetes.io/docs/concepts/services-networking/service/

[16] GitHub Actions, https://docs.github.com/en/actions

[17] GitHub Packages, https://docs.github.com/en/packages

[18] GitLab CI, https://docs.gitlab.com/ee/ci/

[19] Google Container Registry, https://cloud.google.com/container-registry

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

RENOVATION OF THE TRIGGER DISTRIBUTION IN CERN'S OPEN ANALOGUE SIGNAL INFORMATION SYSTEM USING WHITE RABBIT

D. Lampridis*, T. Gingold, D. Michalik[1], T. Pereira da Silva,
A. Poscia, M. H. Serans, M. R. Shukla, CERN, Geneva, Switzerland
[1]also at Aalborg University, Aalborg, Denmark

Abstract

The Open Analogue Signal Information System (OASIS) acts as a distributed oscilloscope system that acquires signals from devices across the CERN accelerator complex and displays them in a convenient, graphical way. Today, the OASIS installation counts over 500 multiplexed digitisers, capable of digitising more than 5000 analogue signals and offers a selection of more than 250 triggers for the acquisitions. These triggers are mostly generated at a single central place and are then distributed by means of a dedicated coaxial cable per digitiser, using a "star" topology. An upgrade is currently under way to renovate this trigger distribution system and migrate it to a White Rabbit (WR) based solution. In this new system, triggers are distributed in the form of Ethernet messages over a WR network, allowing for better scalability, higher time-stamping precision, trigger latency compensation and improved robustness. This paper discusses the new OASIS trigger distribution architecture, including hardware, drivers, front-end, server and application-tier software. It then provides results from preliminary tests in laboratory installations.

INTRODUCTION

A common need among operators in the European Organisation for Nuclear Research (CERN) is to be able to observe, monitor and record the behaviour of equipment in the accelerator complex. Frequently, this behaviour is represented by analogue electrical signals which mirror physical phenomena, such as the current going through a magnet, the magnetic field, the horizontal and vertical position of the particle beam, and so on. Quite often, there is also the need to correlate these measurements, as for example in the case of transfer lines between accelerators where one might want to observe the currents going into the kicker magnets that are used to displace the beam, together with the beam position monitors detecting the beam passing to the next accelerator.

The Open Analogue Signal Information System (OASIS) [1–3] satisfies all of the above requirements by providing an oscilloscope-like user interface, which offers the possibility to select among more than 5000 available analogue signals, acquired from more than 500 multiplexed devices all across the CERN accelerator complex. This functionality implies the existence of a common trigger source, shared by all acquisition devices participating in a given measurement.

In fact, operators are interested in being able to trigger on different conditions, meaning that there is more than one trigger that needs to reach the acquisition devices. Today, OASIS counts over 250 such triggers which are mostly generated at a single central place, where they are also multiplexed. They are then distributed by means of a dedicated coaxial cable per digitiser, using a "star" topology.

This trigger distribution scheme has served OASIS well for many years, but it is beginning to show its limitations. Trigger multiplexers are located close to the central locations where the triggers are generated, while the connection between the multiplexers and the digitisers is done with long coaxial cables, especially since the digitisers are placed close to the analogue signal sources to preserve the integrity of the signals. Furthermore, there is no compensation for cable length differences, environmental conditions, etc. The fact that all triggers are generated at the same building creates a single point of failure. Last but not least, introducing new digitisers typically involves finding a free trigger multiplexer output and pulling the trigger cable from there to the digitiser, an expensive and not always feasible operation.

This paper presents a new trigger distribution architecture for OASIS based on White Rabbit Trigger Distribution (WRTD) [4, 5]. In this new system, triggers are distributed in the form of Ethernet messages over a White Rabbit (WR) [6,7] network, allowing for better scalability, higher time-stamping precision, trigger latency compensation and improved robustness.

BACKGROUND

OASIS

OASIS is built using a 3-tier architecture:

1. The front-end tier controls the hardware modules (digitisers, multiplexers, etc.) and provides a hardware independent interface to the upper layer. It uses the Front-End Software Architecture (FESA) [8, 9] framework to provide the interface to the server tier.

2. The server tier consists of an application server that manages the resources provided by the front-end layer and assigns them to the connections requested by the clients (the third tier). The server tier intends to maximise the number of concurrent acquisitions based on a sophisticated priority algorithm.

3. The application tier, provides the users with a graphical user interface that displays the signals and their settings, in a familiar oscilloscope-like way.

* dimitrios.lampridis@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 1: Typical signal monitoring session in OASIS.

OASIS provides the virtual oscilloscope abstraction. A virtual scope is a software oscilloscope that takes its data from different hardware oscilloscopes and displays it as if it all came from the same module. Thanks to this scheme, users are able to observe several signals distributed around the accelerator complex, as if they were next to each other. Figure 1 shows a typical signal monitoring session in OASIS, with seven signals combined in one virtual scope, triggered by the same trigger.

In order to be able to combine signals coming from different digitisers in a common virtual scope, the hardware needs to be triggered by the same pulse. Figure 2a shows the current trigger generation and distribution scheme used in OASIS. The yellow part on the left represents the central timing system, where timing events are distributed from the master devices (MTT) to numerous timing receivers (CTR) spread across the accelerator complex. The timing receivers in turn are programmed to generate trigger pulses upon reception of a particular timing event, with optional delays and resynchronisation to external clock(s). Once the triggers are generated, they are driven into the inputs of trigger multiplexers (CTC, in green), which also provide the possibility to delay and resynchronise the signal to external clock(s), separately for each output. Finally, each output of a trigger multiplexer is connected to the external trigger input of a digitiser (DAQ, in purple).

WRTD

White Rabbit is a fully deterministic Ethernet-based network which provides sub-nanosecond accuracy. WR is based on the IEEE 1588 Precision Time Protocol (PTP) [10]. The improvements to PTP introduced by WR have been standardised as the High-Accuracy (HA) profile within the IEEE 1588 standard. WRTD provides a generic event distribution system on top of a standard WR network.

WRTD is a generalisation of a previous CERN development, the LHC Instability Trigger (LIST) [11, 12] distribution project. It is modelled after a group of so-called extended functions defined in the Local Area Network (LAN) eXtensions for Instrumentation (LXI) [13], namely the Clock Synchronisation [14], Event Messaging [15] and Timestamped Data [16] functions. The provided Application Programming Interface (API) mimics that of an Interchangeable Virtual Instruments (IVI) [17] driver, with a strong influence from the IVI-3.2 "Inherent Capabilities" [18] and IVI-3.15 "IviLxiSync" [19] specifications.

In WRTD, source nodes receive incoming pulses and distribute them to other nodes over WR in the form of network messages that carry the timestamp of the incoming pulse. The receiving (listening) nodes are programmed to take action (e.g. generate a pulse) upon reception of a particular message, potentially with some fixed delay added to the timestamp.

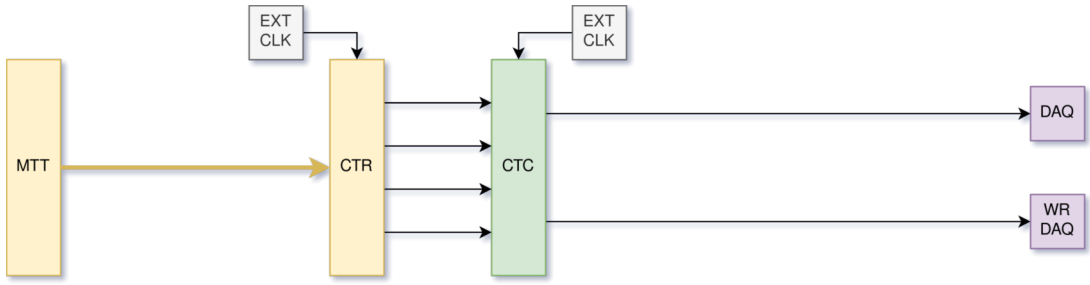
SYSTEM ARCHITECTURE

The proposed solution makes use of WRTD to distribute OASIS triggers over WR and deliver them to the various digitisers. In this system, when a source node receives an incoming trigger pulse, it precisely timestamps its rising edge and distributes it over the WR network to all other nodes, in the form of a timestamp, accompanied by the unique identifier of the source node itself. All nodes that receive the network message and are actively listening for this identifier will add a fixed delay to the received timestamp and configure themselves to generate an output pulse at that precise moment. As long as the fixed delay added is greater than the upper-bound latency of the network (a fundamental feature of WR itself), this precise moment will be in the future and all output pulses will be produced at the same time, thanks to the sub-nanosecond synchronisation provided by WR.

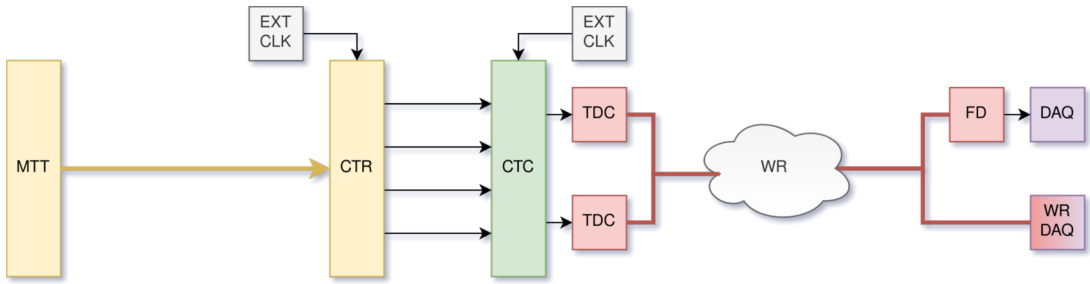
Even though the above scheme will have all digitisers triggering at the same time, that time will still be in the future with respect to the timestamp that was recorded by WRTD at the source node. The solution is to modify the acquisition parameters of the digitisers, in order to have them record enough additional pre-trigger samples to cover the fixed delay that was added to the trigger timestamp, provided of course that the hardware has a large enough memory buffer. This last limitation is only an issue for some old OASIS digitisers with fast sampling rates and small memory and a new project has been launched to design a new equivalent digitiser [20]. Once the acquisition is complete, the acquisition window is rolled back around the original timestamp and the extra post-trigger samples are discarded.

Figure 2b presents a high-level view of the implemented architecture; Compared to Fig. 2a, triggers are still generated by the central timing system (in yellow) and go through the trigger multiplexer (in green), but the outputs of the multiplexer are now timestamped by WRTD source nodes, featuring Time-to-Digital Converters (TDC), which then distribute the triggers as messages over the WR network. At the receiving end, WRTD listening nodes equipped with Fine Delay pulse generators (FD) receive the messages, add

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01



(a) Current scheme



(b) WRTD-based scheme

Figure 2: OASIS trigger distribution architectures.

the globally configured fixed delay and generate the output pulses that are then delivered to the external trigger input of the digitisers. In the special case of a WRTD-enabled digitiser, the FD is omitted since the digitiser is capable of triggering itself by retrieving the WRTD timestamp, taking into account the globally configured delay, and configuring itself to trigger internally at the specified time.

In the coming sections we discuss in more detail the developments performed at the various tiers of OASIS to implement this architecture.

Hardware

The OASIS WRTD network infrastructure is built using standard WR network switches [21] and optical fibres. For the purpose of this project, three new types of WRTD nodes were developed, all of them reusing the WR-enabled Versa Module Europa (VME) dual Field-Programmable Gate Array (FPGA) Mezzanine Card (FMC) carrier, also known as Simple VME FMC Carrier (SVEC) [22] and combining it with existing FMC modules. The most important development in this case was the creation of new FPGA designs (also known as gateware), to provide the necessary WRTD-related logic.

The first WRTD node is a SVEC with the two FMC slots populated with TDC mezzanine cards [23] ("TDC" block in Fig. 2b). Since each TDC mezzanine provides five input channels, the whole WRTD node offers 10 input channels, capable of timestamping incoming pulses and transmitting WRTD events on the WR network.

The second WRTD node is a SVEC with the two FMC slots populated with FD mezzanine cards [24] ("FD" block in Fig. 2b). Since each FD mezzanine provides four output channels, the whole WRTD node offers 8 output channels, capable of generating trigger pulses locally, at precise moments defined in WRTD events received from the WR network.

The third type of WRTD node is a SVEC with the two FMC slots populated with four-channel, 14-bit, 100 MHz ADC mezzanine cards [25] ("WR DAQ" block in Fig. 2b). This is an update to an existing piece of hardware, already used in OASIS, to add the possibility to trigger the ADC directly from WRTD events received from the WR network.

Front-End Tier

FECs related to accelerator controls at CERN run real-time software developed with the FESA framework. Device controls are implemented in FESA classes and instantiated on each FEC, one instance per controlled device. Such an instance is also called a FESA device.

For the renovation of the OASIS trigger distribution system, two new FESA classes were introduced, the WRTD Event Source and the WRTD Event Listener. Even though the design is made with OASIS triggering in mind, the aim was to make it generic enough to be reused in other applications as well in the future. This is why the two new FESA classes do not include "OASIS" in their names and the use of the word "Event" is preferred over that of "Trigger".

Additionally, two existing FESA classes, the OASIS Scope and OASIS Channel were extended.

Global Delay Configuration A global configuration delay value is stored in a configuration file and is made available to all WRTD Event Listener and OASIS Scope and Channel devices.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The purpose of this delay is to make sure that the trigger message is delivered to all WRTD Event Listener devices before the time to produce the trigger at the destination expires. As such, this value must always be superior to the worst-case trigger distribution latency (in the order of 100 µs), which is guaranteed by the upper-bound latency of the WR network itself.

WRTD Event Source The WRTD Event Source FESA class is used to model devices that receive pulses locally, timestamp them and generate messages (Events) over the WR network containing a unique identifier for this device and the timestamp of the pulse. Typically, such devices are individual channels of a TDC.

Each WRTD Event Source device represents one input channel. Thus, a five-channel TDC (which from the hardware point of view is a single device with five input channels) will be mapped to five separate WRTD Event Source devices.

More often than not, the inputs of the TDC will be connected to the outputs of a trigger multiplexer. The trigger multiplexer will be generating the trigger pulses, which will be then timestamped by channels of the TDC. In a few cases, a trigger signal might be directly connected to an input of the TDC, without a trigger multiplexer in between.

WRTD Event Listener The WRTD Event Listener FESA class is used to model devices that receive over the WR network the messages (Events) generated by WRTD Event Source devices and generate pulses locally at a specific moment with respect to the timestamp contained in the received message. Typically such devices are channels of an FD.

Each WRTD Event Listener device represents one output channel. Thus, a four-channel FD (which from the hardware point of view is a single device with four output channels) will be mapped to four separate WRTD Event Listener devices. In practice, the outputs of the FD will be connected to external trigger inputs of digitisers, in order to trigger the acquisition of the digitisers at the moment specified by WRTD.

Apart from FD generators, another type of device that is modelled by the WRTD Event Listener is a digitiser that is capable of triggering directly over WRTD. Such a digitiser will also provide a single WRTD Event Listener to model its capability to receive the WRTD message and produce the local trigger pulse internally.

OASIS Scope and Channel The OASIS representation of a digitiser is based on the combination of two FESA device classes, the OASIS Scope and Channel. The first one handles all aspects of triggering and "horizontal settings" (sampling rate, number of samples, etc.), while the second one handles the "vertical settings" (full-scale range, offset, etc.).

In the context of this project, these two FESA classes have been extended, to allow for the triggering over WR. In particular, the following modifications were performed:

1. A new configuration option was introduced to specify whether this scope is triggered over WR or not.

2. If a scope is triggered over WR, the global delay configuration is taken into account, affecting the number of pre-trigger acquired samples, in order to allow for the trigger latency introduced by the WRTD network.

3. Once such a scope is triggered, the global delay configuration value is subtracted from the trigger timestamp, in order to point to the moment in time when the original trigger was generated at the source, not when it was actually received by the digitiser.

4. Following that, the acquisition window is adjusted once again, in order to discard surplus samples and return the data around the original trigger moment.

It is worth noting that these calculations and adjustments of settings happen at the front-end tier level. The OASIS server and the applications are not aware of them. Thus, if a user selects a particular trigger and asks for 1000 samples with 50% pre-triggering, the front-end tier will silently perform the necessary adjustments and in the end it will simply return 500 samples before and 500 samples after the selected trigger.

Configuration Database

OASIS relies heavily on relations defined in the Controls Configuration Database (CCDB), to associate signals to signal multiplexers and digitiser channels, triggers to trigger multiplexers and digitisers, and so on.

In the context of OASIS WRTD, three new relations have been introduced and one existing relation has been reused:

COUNTER2WRTD: This new relation connects the output of a trigger multiplexer to a WRTD Event Source.

TRIGGER2WRTD: This new relation connects a trigger signal directly to a WRTD Event Source.

WRTD2SCOPE: This new relation connects a WRTD Event Listener to a digitiser.

TRIGGER2MUX: This existing relation connects a trigger signal to the input of a trigger multiplexer.

Server Tier

For the OASIS server to be able to route a specific trigger signal to a digitiser, it needs to identify a route and then to configure the devices on that route. The server discovers that information using the relations defined in CCDB.

WRTD Route Identification To find a possible route for the trigger signal, the server first needs to discover which WRTD Event Source devices the trigger signal is connected to.

The actual route identification requires the following steps:

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

1. Discover to which trigger multiplexer(s) and on which input the trigger signal is connected (via a TRIGGER2MUX relation).

2. Identify all WRTD Event Sources connected to the trigger multiplexer(s) found in (1), remember to which output of the multiplexer each WRTD Event Source is connected (via a COUNTER2WRTD relation).

3. Alternatively, detect whether the trigger signal is directly connected to a WRTD Event Source, without any trigger multiplexer in between (via a TRIGGER2WRTD relation).

Thanks to this initial discovery phase, a set of candidate WRTD Event Sources to use to distribute the trigger on the WR network is obtained. Among those candidates, the server then finds the first one that matches all of the following criteria:

1. The WRTD Event Source is running and reachable.

2. The WRTD Event Source is free (i.e. it is not already used to route a trigger).

Once a WRTD Event Source is identified, the server queries the declared CCDB relations to find out to which WRTD Event Listener the digitiser is connected to.

At this point all the equipment that will be used to route the trigger signal has been identified. This includes:

- The WRTD Event Source
- The trigger multiplexer and the input and output to use
- The WRTD Event Listener

WRTD Route Configuration After a route is identified, it is necessary to configure the different devices so that the trigger signal is correctly routed to the digitisers. The following operations are performed by the OASIS server:

1. Trigger multiplexer: close the route from the trigger signal to the selected WRTD Event Source

2. WRTD Event Source: enable the device

3. WRTD Event Listener: enable the device and set the WRTD event to be listened to

Following these operations, the trigger signal will be routed to the WRTD Event Source that will publish the trigger to the WRTD network. The WRTD Event Listener will listen to the event published by that source device and trigger the digitiser accordingly. Finally, when the user has finished observing the signal(s), the OASIS server needs to turn off the WRTD Event Source and Listener devices and release the route that was used in the trigger multiplexer.

Application Tier

Even though there are several applications that make use of OASIS, the most widely used one is the OASIS viewer (also shown in Fig. 1).

One of the core decisions taken during the planning phase of this project was to avoid any significant modifications and user interface redesigns of the OASIS viewer. Users of OASIS, CERN operators in particular, are accustomed to the current interface and the idea is to first renovate the trigger distribution system and then, slowly start introducing new features that would require an interface redesign, such as the possibility for a digitiser to trigger other digitisers, something that is not available with today's unidirectional trigger distribution system.

As such, the only modification that was done to the OASIS viewer was to add more information regarding the configured trigger path in the case of WRTD triggers (that is, the device names of the WRTD Event Source and Listener used).

TEST SETUP AND FIRST RESULTS

A standard CERN VME FEC with all the necessary hardware was deployed in the laboratory from the beginning of this project. This test setup was used extensively during development, also as a part of a Continuous Integration (CI) flow. Currently, this setup is running released and deployed versions of all components and it is being used for the final validation of the full system.

A high-level diagram of the test setup can be seen in Fig. 3. Two trigger pulses are generated by the timing system, separated by a 400 µs delay. Both triggers are fed into a trigger multiplexer. At the same time, the two trigger signals are also connected to analogue input channels of two scopes; the "normal" trigger goes to channel 1 of both scopes and the delayed trigger to channel 2.

Scope 2 is triggered directly by one of the outputs of the trigger multiplexer. Another output of the same trigger multiplexer is connected to a WRTD Event Source, and a WRTD Event Listener output is connected to the trigger input of Scope 1. Thanks to the trigger multiplexer, each scope can be triggered on either of the two signals, independently of each other. The WR network itself is made of two WR switches, separated by 10 km of rolled optical fibre. The global delay is configured at 400 µs.

Figure 4 shows the OASIS viewer application, with the four signals connected. The yellow traces are from Scope 1, while the magenta traces are from Scope 2. The OASIS viewer is configured with two virtual scopes; the top one displays the signals from channel 1 of both scopes, while the bottom one shows the channel 2 signals. Both of the scopes are set to trigger on the signal that is also connected to channel 1. The acquisition window is set to 500 µs (50 µs/div) and there is a horizontal delay of 200 µs applied. As it can be seen, there is no visible difference between the scope that is triggered by the pulse itself, and the scope that is triggered by the pulse passing first through the WRTD network, despite the latency of the network; the signals on both virtual

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

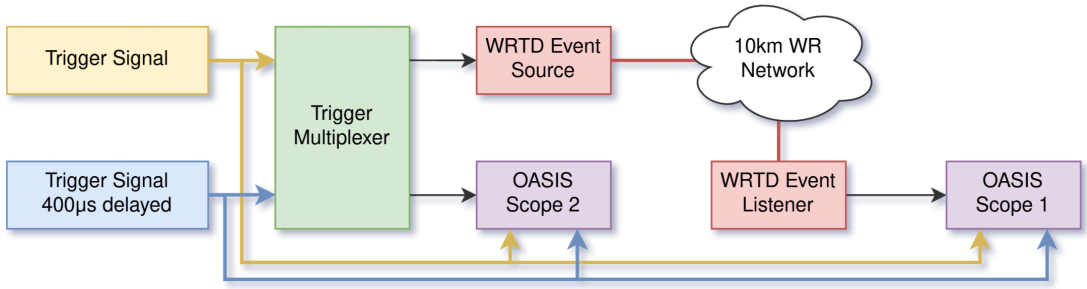


Figure 3: Laboratory test setup.

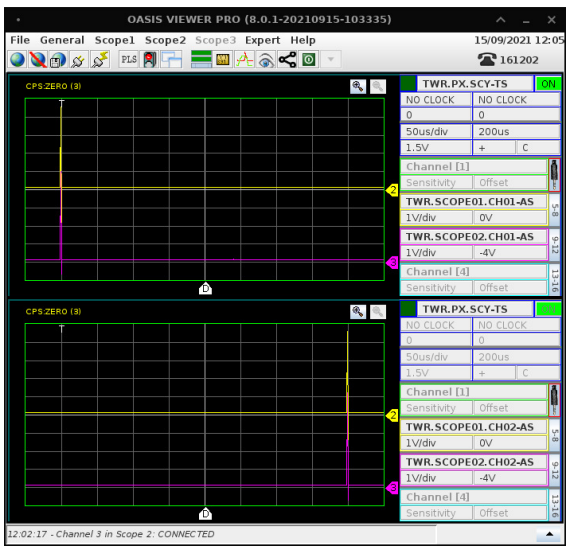


Figure 4: OASIS signal acquisition with WR triggers.

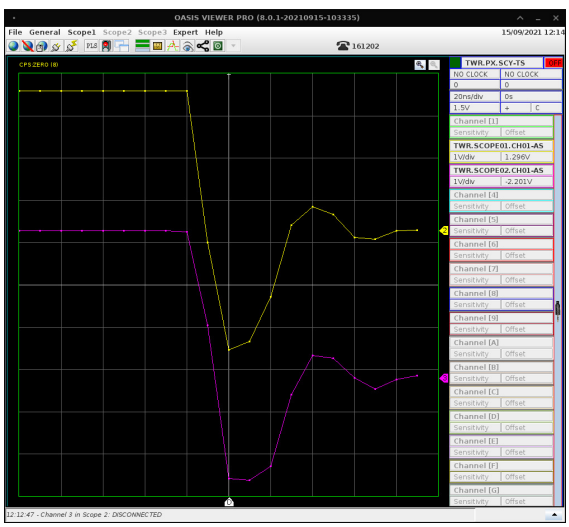


Figure 5: Zoom at trigger moment.

scopes are fully aligned, with the two sets of signals being 400 µs apart.

Figure 5 is a zoom in the moment of the trigger, using the two channel 1 signals from both scopes. The two digitisers are acquiring with a 100 MHz sampling rate (10 ns sampling period) and the OASIS viewer is set to display a 200 ns window. As a result, one can see the 20 individual samples from each scope (each sample represented by a dot) and how they are aligned to each other, despite the fact that the two scopes are triggered in two completely different ways.

CONCLUSIONS AND OUTLOOK

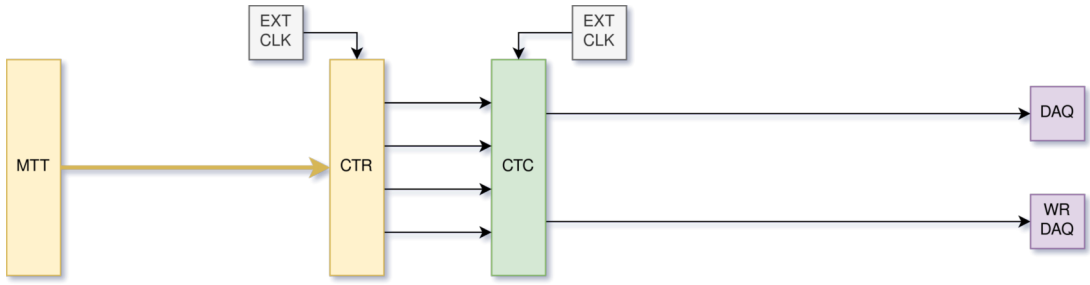
This paper has presented the new OASIS trigger distribution architecture based on WRTD. Currently, the development effort is almost complete, and the laboratory tests show that the full hardware and software stack is working and is well integrated inside the CERN accelerators' control system.

It is expected that the first operational OASIS triggers over WR will circulate before the end of 2021. Following that, a large campaign will take place to start replacing the complete trigger distribution installation.

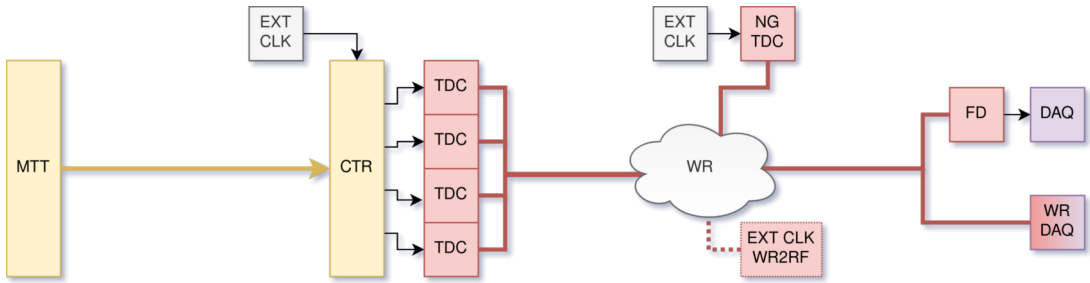
There are also plans to further evolve the system, in three successive phases, with the work presented in this paper being part of the first phase. Figure 6 shows the proposed road-map, including a repetition of the block diagram from Fig. 2b to make the comparison between the three phases easier. In particular, the second phase involves the complete elimination of trigger multiplexers, in favour of direct timestamping of triggers by TDC channels. For this to happen however, new hardware developments are necessary, in order to provide a way to synchronise WRTD timestamps to external clocks, a feature provided today by the trigger multiplexer. This could happen for example with a new generation of TDC (NG-TDC in Fig. 6b), or with a module capable of receiving and reconstructing clock signals over a WR network (such as the WR2RF module [26] in Fig. 6b). In the third and last phase, that requires the complete central timing to be migrated to WR, there will be no more trigger pulses to digitise, and OASIS will be picking up timing events/messages from the WR network and using them for triggering its digitisers.

On another front, since WRTD itself is based on the existing IVI/LXI standards, it is foreseen to try to merge WRTD with these standards, in the same way that WR was merged and standardised within the IEEE1588 PTP standard. Such a development would allow instruments with an IVI driver and a WR interface to exchange events with each other. This

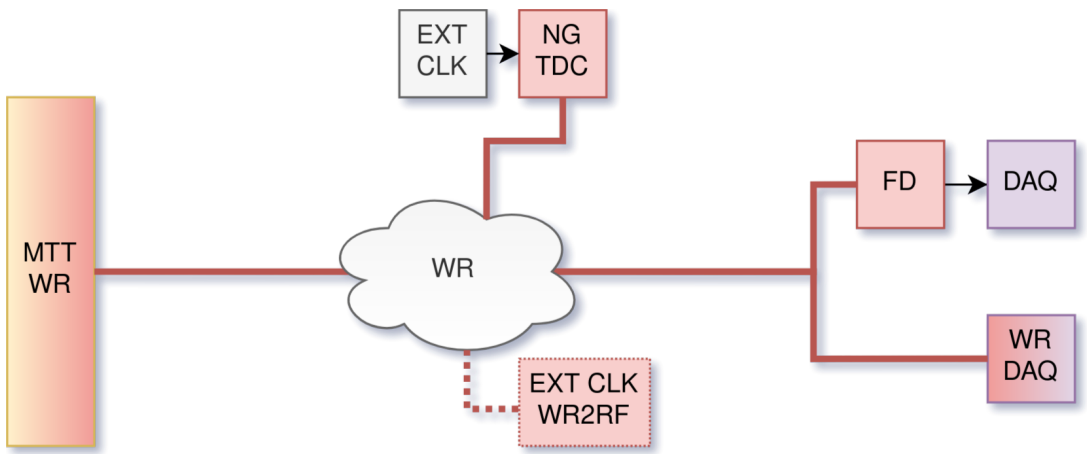
ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01



(a) OASIS WRTD: phase 1



(b) OASIS WRTD: phase 2



(c) OASIS WRTD: phase 3

Figure 6: OASIS trigger distribution evolution road-map.

would also allow the industry to produce commercially available WRTD-enabled devices, which OASIS could then procure and use out of the box, with minimum integration effort.

REFERENCES

[1] S. Deghaye, D. Jacquet, I. Kozsar, and J. Serrano, "OASIS: a New System to Acquire and Display the Analog Signals for LHC", in *Proc. 9th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'03)*, Gyeongju, Korea, Oct. 2003, paper WP502, pp. 359–361.

[2] S. Deghaye, L. Bojtar, Y. Georgievskiy, and J. Serrano, "OASIS Status Report", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'05)*, Geneva, Switzerland, Oct. 2005, paper TH3A.1-5O.

[3] L. Bojtar, C. Charrondiere, Y. A. Georgievskiy, F. C. Peters, I. S. Zharinov, and S. Deghaye, "OASIS Evolution", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper WPPA04, pp. 322–324.

[4] D. Lampridis, T. Gingold, M. Malczak, F. Vaga, A. Wujek, and T. Włostowski, "The Distributed Oscilloscope: A Large-Scale Fully Synchronised Data Acquisition System Over White Rabbit", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 725–732. doi:10.18429/JACoW-ICALEPCS2019-TUBPR01

[5] *WRTD* project page, https://ohwr.org/project/wrtd/wikis/home

[6] *White Rabbit* project page, https://ohwr.org/project/white-rabbit/wikis/home

[7] J. Serrano *et al.*, "The White Rabbit Project", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Control Systems (ICALEPCS'09), Kobe, Japan, Oct. 2009, paper TUC004, pp. 93–95.

[8] L. Fernandez *et al.*, "Front-End Software Architecture", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.

[9] A. Schwinn *et al.*, "FESA3 – The New Front-End Software Framework at CERN and the FAIR Facility", in *Proc. 8th Int. Workshop on Personal Computers and Particle Accelerator Controls (PCaPAC'10)*, Saskatoon, Canada, Oct. 2010, paper WECOAA03, pp. 22–26.

[10] *IEEE 1588 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Institute of Electrical and Electronics Engineers, 2008, ISBN 978-0-7381-5400-8.

[11] *LIST* project page, `https://ohwr.org/project/list/wikis/home`

[12] T. Wlostowski *et al.*, "Trigger and RF Distribution Using White Rabbit", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, 2015, paper WEC3O01, pp. 619–623.

[13] LXI Consortium, `http://www.lxistandard.org`

[14] *LXI Clock Synchronization Extended Function*, LXI Consortium, revision 1.0, 8 November 2016.

[15] *LXI Event Messaging Extended Function*, LXI Consortium, revision 1.0, 8 November 2016.

[16] *LXI Timestamped Data Extended Function*, LXI Consortium, revision 1.0, 8 November 2016.

[17] IVI Foundation, `http://www.ivifoundation.org`

[18] *IVI-3.2: Inherent Capabilities Specification*, IVI Foundation, revision 2.1, 9 March 2015.

[19] *IVI-3.15: IviLxiSync Specification*, IVI Foundation, revision 2.0, 9 October 2014.

[20] *Fmc-Adc-1G-8b-2cha* project page, `https://ohwr.org/project/fmc-adc-1g8b2cha/wikis/home`

[21] *White Rabbit Switch* project page, `https://ohwr.org/project/white-rabbit/wikis/Switch`

[22] *SVEC* project page, `https://ohwr.org/project/svec/wikis/home`

[23] *Fmc-Tdc-1ns-5cha* project page, `https://ohwr.org/project/fmc-tdc-1ns-5cha-gw/wikis/home`

[24] *Fmc-Del-1ns-4cha* project page, `https://ohwr.org/project/fmc-delay-1ns-8cha/wikis/home`

[25] *Fmc-Adc-100M-14b-4cha* project page, `https://ohwr.org/project/fmc-adc-100m14b4cha/wikis/home`

[26] *WR2RF-VME* project page, `https://ohwr.org/project/wr2rf-vme/wikis/home`

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

WHITE RABBIT AND MTCA.4 USE IN THE LLRF UPGRADE FOR CERN'S SPS

Tomasz Włostowski*, Andrew Butterworth, Grzegorz Daniluk,
Julien Egli, John Robert Gill, Tristan Gingold,
Juan David González Cobas, Michel Arruat, Grégoire Hagmann, Dimitrios Lampridis,
Maciej Marek Lipiński, Mattia Rizzi, Arthur Spierer,
Maciej Sumiński, Adam Artur Wujek, Karol Adrianek,
Predrag Kuzmanović, Philippe Baudrenghien, Saúl Novel González,
Julien Palluel, CERN, Geneva, Switzerland

Abstract

The Super Proton Synchrotron (SPS) Low-level RF (LLRF) system at CERN was completely revamped in 2020 [1]. In the old system, the digital signal processing was clocked by a submultiple of the RF. The new system uses a fixed-frequency clock derived from White Rabbit (WR) [2]. This triggered the development of an eRTM module for generating very precise clock signals to be fed to the optional RF backplane in MTCA.4 crates. The eRTM14/15 sandwich of modules implements a WR node delivering clock signals with a jitter below 100 fs. WR-clocked RF synthesis inside the FPGA makes it simple to reproduce the RF elsewhere by broadcasting the frequency-tuning words over the WR network itself. These words are received by the WR2RF-VME module and used to produce beam-synchronous signals such as the bunch clock and the revolution tick. This paper explains the general architecture of this new LLRF system, highlighting the role of WR-based synchronization. It then goes on to describe the hardware and gateware designs for both modules, along with their supporting software. A recount of our experience with the deployment of the MTCA.4 platform is also provided.

INTRODUCTION

The High Luminosity LHC (HL-LHC) project at CERN aims to increase the integrated luminosity of the LHC by a factor of 10 in the 10-12 years after its implementation in 2027. As a result of new requirements for the LHC beam, the whole CERN injector complex has undergone an extensive upgrade program. For the SPS, the synchrotron just upstream of the LHC, the new requirement for a beam intensity of 2.3×10^{11} protons/bunch with a bunch spacing of 25 ns resulted in a need to overhaul the complete accelerating system, including cavities, amplifiers and LLRF.

The new LLRF system needed to be ready in a short amount of time and required a significantly higher data transfer rate from the cards to the host PC than the legacy VME platform. A decision was therefore taken to capitalize on the effort of DESY and other institutes, as well as the Commercial-Off-The-Shelf (COTS) components using the MTCA.4 standard. Field Programmable Gate Arrays (FPGA) and System-on-Chip (SoC) solutions were used extensively to speed up the development and provide the flexibility allowing the system to be reused in future applications.

White Rabbit (WR) timing technology plays a key role in several parts of the system:

- It is used to derive a very low-noise base clock signal which is then used to sample the cavity antenna and beam Pick-Up signals, and to synthesize the RF drive signals.
- It is the transmission medium through which the LLRF system receives bending magnetic field information in real time.
- Coupled with Direct Digital Synthesis (DDS) technology, it allows the LLRF system to transmit the RF signals not as physical signals but as messages containing Frequency-Tuning Words (FTW) which allow receivers to replay the RF in synchronism all around the accelerator. This allows distributing the reference RF phase with fixed and very stable latency.

This paper does not aim at describing the upgraded LLRF system in general, but just the synchronization aspects and the use of the WR technology therein. After a quick introduction to the general architecture of the system, we describe the low-noise eRTM14/15 clock signal generation and distribution module. We then move on to the module allowing regeneration of RF signals based on reception of FTWs, the WR2RF-VME. Finally, we share our experience as new users of the MTCA.4 form factor, hoping it can be useful to others in their exploration for optimal platforms to implement LLRF or other types of systems.

GENERAL ARCHITECTURE

The overview of the SPS LLRF system architecture is shown in Fig. 1. Note that it focuses on the usage of WR in the system and omits many details of the RF part.

The renovated system drives six 200 MHz RF cavities and is implemented entirely in the MTCA.4 form factor. Each cavity has a dedicated Cavity Controller (CC), consisting of a Struck SIS8300KU card [3] and a DS8VM1 analog frontend/vector modulator RTM from DESY [4]. The CC's primary role is to maintain stable cavity power and phase, compensating the error introduced by the power amplifier (the Polar Loop) as well as handling the cavity beam loading

* Tomasz.Wlostowski@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

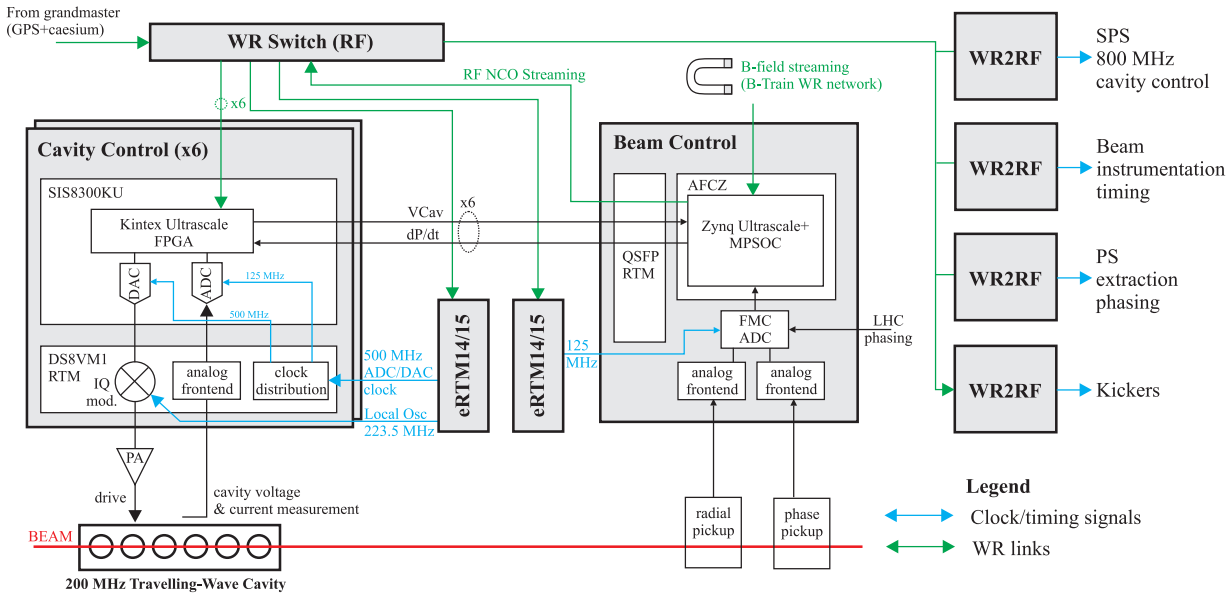


Figure 1: Overview of the WR-based SPS Low Level RF system.

and impedance effects (One-Turn Feedback). The measured cavity voltage information is streamed to the Beam Controller (BC) over dedicated gigabit links. Furthermore, the CC implements a monitoring and protection system which ensures safe operation of the high power RF equipment.

The BC uses the AFCZ board by Creotech [5] and two custom-made FMC mezzanines to interface with the power plant and the beam instrumentation. Its main task is calculating the instantaneous RF frequency and cavity voltage setpoints for a particular beam (particle type, intensity and fill pattern) during the acceleration ramp. The most significant inputs for the BC are the beam's transverse position, phase, current and the magnetic field in the machine's magnets. The latter is provided by a WR-based B-Train system [6], also using a dedicated WR link. The frequency information calculated by the BC is broadcast with every beam turn over the WR network using a protocol similar to the B-Train system. More details are provided in the section about the WR2RF-VME module.

Timing-wise, each CC requires two clock signals: the Local Oscillator for upconversion of the RF drive output (223.5 MHz sine wave) and a 500 MHz ADC/DAC clock. Both clocks have stringent phase noise (PN) and stability requirements: PN of −130 dBc/Hz at 1 kHz (carrier 223.5 MHz) and precision better than 13 ps (1 degree at 200 MHz). These requirements were set to minimize the detrimental effect of the RF noise during the long filling plateau for the LHC ions beams in the SPS. The clocks are provided by the eRTM14/15 module, described later. The BC needs a 125 MHz WR-synchronous clock for its ADCs, with similar noise and stability figures as the CC.

All six CCs share a single MTCA.4 crate (Schroff/nVent model 11890-170) with a single MCH/CPU set from N.A.T. [7] and a single eRTM14/15 module, whereas the BC resides in a separate MTCA.4 crate with its own CPU/MCH/eRTM boards. The crates are also equipped with the CERN General Machine Timing receiver cards (not included in the drawing for the sake of clarity). The low-jitter clock connections between the eRTM board and the CCs are provided by the DESY MTCA.4 LLRF backplane (RFBP) [8].

As the new LLRF system distributes the frequency information in a purely digital form, a need arises for interfacing with the legacy systems (mostly VME-based) that require an analog cavity frequency and RF-related trigger signals (e.g. kicker magnets, beam instrumentation, Proton Synchrotron (PS) extraction synchro and the auxillary 800 MHz SPS cavities). This task is performed by the WR2RF-VME cards, described in detail in the subsequent chapters.

THE ERTM14/15 MODULES

The eRTM14/15 [9] is a MTCA.4-compliant WR receiver providing a variety of ultra-low jitter clock signals which are distributed over the RFBP:

- Two DDS clocks (named LO and REF in this paper), distributed to slots 4..12 of the MTCA crate and with two front-panel outputs for up/down-modulation of the RF analog signals.
- Two digital clocks (distributed to slots 4..12 and the front panel).
- 10 MHz and Pulse Per Second (PPS) input and output for interfacing with non-WR timing devices.

The module is made of two boards connected with a high speed board-to-board cable (Samtec QSH series), as depicted in Fig. 2. The board residing in slot 14 contains the FPGA (Kintex-7 XC7K70T) that implements the WR synchronization stack (the WR PTP Core - WRPC [10]), two redundant SFP uplink ports and basic ("low performance") WR local oscillators. Slot 15 contains the high performance oscillators and analog circuitry synthesizing the clocks listed above. Therefore, the eRTM14 can work alone as a basic

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

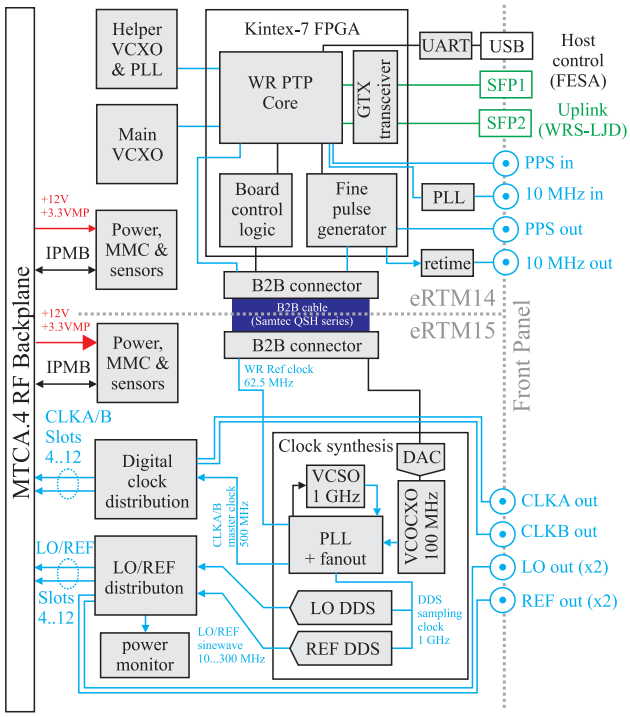


Figure 2: Simplified block diagram of the eRTM14/15 WR Receiver.

WR receiver or together with the eRTM15 with the full set of features.

To achieve the timing performance required by the SPS LLRF system, we had to develop novel techniques described in the two next subsections.

WR Clock Synthesis

The WR clock recovery and synthesis was designed to meet a −130 dBc/Hz phase noise figure at 1 kHz offset (carrier frequency 223 MHz) and less than 100 fs rms jitter (100 Hz - 10 MHz).

Meeting the latter requirement necessitated a local oscillator with sufficiently low PN in the low frequency range to match −100 dBc/Hz, which is the noise floor of the Digital Dual Mixer Time Difference (DDMTD) phase detector used by WR at the offset of 50 Hz, much less than the bandwidth of standard WR PLL implementations (about 300 Hz). An Axtal AXIOM45 series OCXO has been selected, as a compromise between the performance and the price.

The OCXO provides a 100 MHz sinewave signal, fine-tuned within the range of a few ppm by a low-noise DAC (AD5660) controlled by the WRPC. This 100 MHz is multiplied by the subsequent PLL synthesizer (LTC6950) that delivers the 62.5 MHz WR reference clock (thus closing the WR clock recovery feedback loop), a 500 MHz master clock for the CLKA/B outputs and a 1 GHz sampling clock for the LO/REF DDS synthesizers. The PLL uses a discrete voltage-controlled SAW oscillator (Crystek CVCSO-914) to further improve the PN in the high frequency range and provide enough headroom for the additive PN of the DDS and clock distribution circuitry.

The LO/REF clocks are produced by two AD9910 integrated DDS synthesizers, capable of outputting up to 400 MHz with a tuning resolution of 0.23 Hz. The eRTM design allows them to be programmed to any arbitrary (non-modulated) frequency between 10 and 300 MHz. The output of each DDS is followed by a two-stage clock distribution network, each stage consisting of a Mini-Circuits GVA-81 amplifier followed by a passive 1:4 splitter. As the distribution network is partly passive, the outputs are coupled to the RFBP with individually programmable MEMS RF switches, terminating the unused slots. Each LO/REF RFBP output is also equipped with an RF power monitor. The module also allows to adjust (globally) the output power of the LO and REF chains between +9 and +19 dBm, achieved by digitally changing the DDS DAC bias current.

Furthermore, the DDS outputs can be phased together at an arbitrary moment of time through a message sent over the WR network. Such a message contains a TAI timestamp at which the Fine Pulse Generator core in the FPGA generates a pulse to reset the AD9910 phase accumulators, thus ensuring all the chips start producing the LO/REF signals at exactly the same phase. The SPS LLRF uses this feature at the start of each beam cycle to ensure the LO's in all WR nodes are correctly aligned.

The CLKA/B clocks are digital, differential (LVPECL levels) intended for driving ADC/DACs. They are derived from a 500 MHz master clock through two LTC6953 clock dividers/fanout buffers. The user can select from the frequencies of 62.5, 125, 250 and 500 MHz, each output capable of independent frequency, phase and on/off setting. To ensure the fixed phase alignment of the divided-down clocks across different nodes in the network, the FPGA's Fine Pulse Generator core provides PPS-aligned divider synchronization pulses for the LTC6953s.

The PN figures, measured for CLKA and LO channels

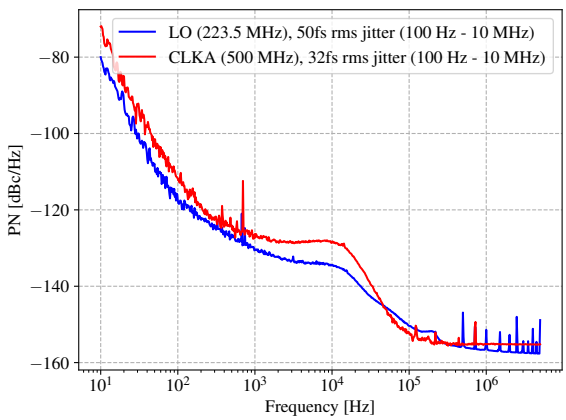


Figure 3: Phase noise of the CLKA (500 MHz) and LO (223.5 MHz) outputs of the eRTM.

Furthermore, for the eRTM to achieve the required performance, the upstream elements of the SPS WR network had to be improved:

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Changing the master GPS receiver to Microchip Syncserver S650 [11], as the previous receiver showed small phase jumps (few dozen ps), sufficient to disrupt the operation of the LLRF system referenced to it.
- Upgrading all the upstream WR Switches [12] to the Low Jitter Daughterboard (WRS-LJD) version [13], which offers lower additive PN and improved DDMTD tracking stability. More details can be found in [14].

Phase Stability

The second requirement of the SPS LLRF that resulted in substantial development effort was the phase stability (precision) better than 13 ps. In other words, every time the system is switched on, the clock phases between all RF nodes must not change by more than 13 ps. Standard WR implementations offer a precision of around 100 ps, which we had identified to come from the internal clocking and Clock/Data Recovery (CDR) of the FPGA's Gigabit transceivers. We focused on the Xilinx's GTXE1 and GTXE2 transceivers, used, respectively, in the Virtex-6 and Kintex-7 series of FPGAs and improved their phase stability to better than 5 ps (std). The excessive phase error is caused by multiple effects:

- The RX/TX buffer bypass logic aligning the SerDes high clock with the FPGA fabric clock, which has been implemented as a tapped delay line, with the tap selected during the initialization of the link.
- Various clock dividers (most importantly, the RX SerDes bit clock to parallel word clock).

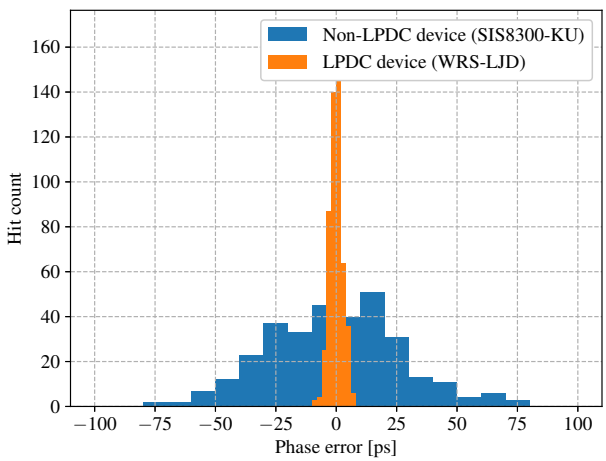


Figure 4: Phase stability comparison between LPDC and standard WR nodes.

These effects have been mitigated by instrumenting the transceiver internal clocks with a DDMTD-based phase measurement unit and repeatedly resetting the transceiver until the clock phases are fixed on previously calibrated values. The detailed information about the procedure is available in [15]. The WR devices which offer such improved phase stability, such as the version 6.0 of the White Rabbit Switch, are marked as Low Phase Drift Calibration (LPDC). Figure 4 compares the phase stability of a standard WR PHY implementation with the LPDC-enabled transceiver of the eRTM board.

Digital and Software

All timing features of the eRTM module are controlled by the Kintex-7 FPGA, encompassing the WRPC, a set of I2C/SPI/GPIO interfaces to control all onboard chips and a Fine Pulse Generator core to produce the PPS output and the counter/phase accumulator synchronization pulses.

The WRPC contains an embedded CPU core (LM32) which runs a customized version of the WRPC software. The key differences between the standard WRPC and the eRTM are:

- Support for eRTM board-specific circuitry (PLLs, DDS, clock distribution).
- Improved regulation loops of the WR PLL (PI gain optimization, gain scheduling).
- Remote control and monitoring of the board through the front panel USB port. A library in C/Python is provided, which allows a remote PC to control the most important run-time parameters of the board (clock frequencies, channel assignment, etc.).

Being an MTCA.4-compliant module, the eRTM implements two MTCA Management MicroControllers (MMCs), one per board. These are STM32F3 series MCUs, running a customized version of the LNLS's OpenMMC firmware [16]. Their key responsibilities are monitoring the board health (voltages, currents and temperatures), power sequencing and MTCA.4 identification of the boards.

THE WR2RF-VME MODULE

The WR2RF-VME card is a VME-compatible WR RF receiver, featuring:

- High performance WR receiver.
- x2 RF channels, each capable of reconstructing the modulated RF signal.
- x2 Programmable trigger outputs per RF channel.
- A number of standard timing I/Os (10 MHz/PPS) and general-purpose digital inputs/outputs.

The simplified block diagram of the WR2RF-VME is depicted in Fig. 5. As the board inherits the clock generation circuitry (OCXO, PLL and DDS) from the eRTM board, we will focus exclusively on the RF-specific features.

RF Streaming and Reconstruction

The SPS Beam Control distributes Ethernet frames known as the "RF-train" over the WR network to Cavity Controllers and WR2RF-VME cards. These frames contain Frequency Tuning Words (FTW) that describe the setpoints for Numerically Controlled Oscillators (NCOs) implemented within the Kintex7 FPGA on the WR2RF-VME card. To ensure all receivers have the same NCO value, two things are necessary:

- Every frame from Beam Control is transmitted using WR Streamers, a feature of the WR-Core that guaran-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

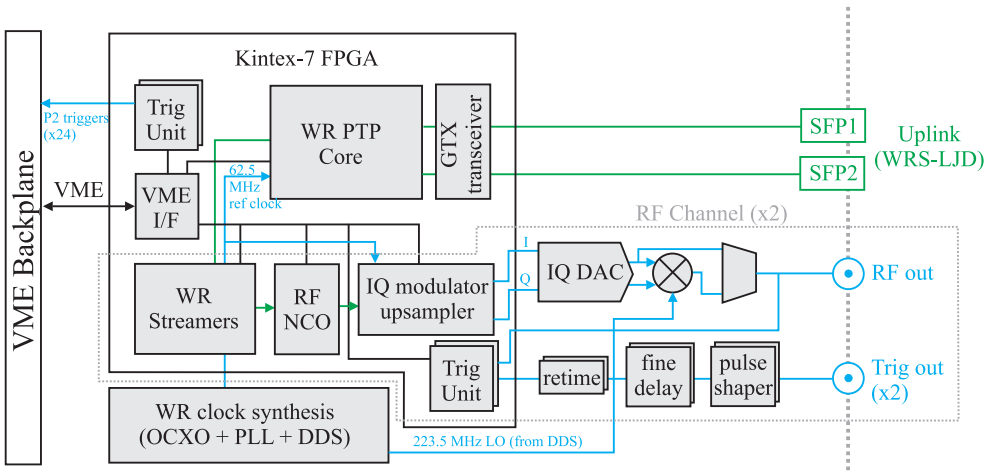


Figure 5: Simplified block diagram of the WR2RF-VME board.

tees a fixed latency delivery (the fixed latency must be larger than the worst case network latency).

- Each NCO is reset prior to starting a new accelerator cycle.

These features ensure that each NCO, in every receiver, has the same value on any WR clock tick.

The RF is reconstructed using a mixer approach. The FPGA uses the output of the NCOs to drive I and Q data lines for an AD9783 Digital to Analog Converter (DAC), with a varying frequency close to 23.5 MHz. The DAC generates the Intermediate Frequency (IF) for the LTC5598 mixer. Meanwhile, the Local Oscillator (LO) input of 223.5 MHz for the mixer is derived from an AD9910 DDS chip, as described for the eRTM modules. The output from the RF mixer is then filtered to accept the downside conversion and attenuate the LO and image frequency leakage close to 200 MHz. This is the reconstructed RF signal.

Trigger Units

In LLRF systems at CERN, a Trigger Unit (TU) [17] is used to generate a pattern of pulses that are synchronized with an RF signal. These TUs, implemented in FPGA logic, are clocked by a digital version of the reconstructed RF signal to create an RF synchronous system. In the WR2RF-VME card there are two cascaded TUs, and typically they create:

- The revolution frequency or orbit clock.
- A bunch clock, which is a divided by 5 digital version of the RF signal.

Applications of these TUs are:

- Precise RF bucket selection to adjust the injection of particle bunches.
- Timing pulses to control systems responisble for dumping the beam.
- Signals for beam instrumention and monitoring.

The output from the TU then drives a discrete pulse shaping and retiming circuit on the WR2RF-VME card, allowing for low jitter trigger signal generation and precise adjustment of trigger delay (10 ps steps).

EXPERIENCE WITH MTCA.4

The SPS LLRF system is the first medium-scale deployment of the MTCA.4 platform in the Accelerator Sector at CERN. While MTCA.4 has easily met the project's requirements regarding the CPU processing power and the PCI Express bandwidth compared to the VME platforms, we identified several issues affecting the reliability and integration of the platform in the CERN Controls framework, in particular the LLRF backplane and the modules which utilize it. The list below applies only to the hardware models used at CERN and not the MTCA.4 ecosystem as a whole:

- Suboptimal thermal design of the MTCA.4 crate and fan speed management: the fans in the front were too powerful with respect to the fans in the rear. As the crate's cooling air inlet is in the front, the stock version of the fan unit would direct almost all of the airflow to the front part of the crate, thus causing overheating of the rear modules such as the Rear Power Modules (RPMs) and the eRTM board. We have observed temperatures of ~ 90 °C on the eRTM at an approximate power dissipation of 36 W (per 2 slots). Furthermore, the eRTM15 slot that dissipates the most heat in our system receives less airflow compared to other RTM slots. This is due to the placement of the fan too far from the slot.
- Unreliable Power Module (PM) firmware. The dedicated RFBP rear PMs would refuse to work with the eRTM module without indicating any error. We continued our development using standard PMs, but the MCH refused to recognize the eRTM boards (no IPMI/I2C communication).
- Bugs in the MCH, such as non-functional "upstream slot power delay" parameter causing PCIe enumeration issues for FPGA-based PCIe cards.
- CPU configuration process. The BIOS/UEFI settings of the CPU module are extremely complex and there is no recovery mechanism ("CMOS reset"). Incorrect settings can render the board permanently inoperable.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

It is worth noting that after an intense, 4-month collaboration with the vendors, the above issues have been fixed or mitigated by the companies, thus making the MTCA.4 platform technically viable for current and future applications in the Accelerator sector at CERN.

On the non-technical side, the SPS renovation project's relatively short timespan required the RF team to choose from COTS MTCA.4 modules instead of the traditional in-house-developed electronics. Unfortunately, to obtain the basic information required to integrate one of the modules, a Non-Disclosure Agreement was needed. This is undesirable in a scientific environment and the long signature process even impacted the project planning.

CONCLUSION

The MTCA.4-based SPS LLRF system has been in commisioning since April 2021 and has operated with physics beams since July 2021. After 6 months of successful commissioning, the results are very positive. We plan to use this novel solution for the prototype upgrade of the HiLumi LHC Accelerating and Crab Cavity LLRF systems.

The key highlights are:

- Timing performance previously reserved for analog RF distribution systems in a WR-network based solution.
- Proven RF distribution over WR, driving accelerating cavities in an operational machine (SPS).
- Compactness: taking two 9U MTCA.4 crates and one 19-inch rack with custom electronics instead of over ten 19-inch racks.
- Backplane connectivity for low-jitter clocks and RF signals reducing cabling cost and complexity.
- Bandwidth of the PCI Express fabric provided by MTCA.4 outperforming the legacy VME platform by two orders of magnitude and allowing for much more sophisticated beam diagnostics.
- Usage of COTS modules significantly shortened the development time (especially with the SIS8300-KU and AFCZ boards).

The SPS LLRF project also triggered the development of ultra-low phase noise and improved precision White Rabbit equipment: the eRTM14/15, WRS-LJD and WR2RF-VME which offer unprecedented performance among the available WR devices. These boards are available under the CERN Open Hardware Licence [18] and can be easily adapted to other LLRF systems as well as beam instrumentation or experiment electronics.

ACKNOWLEDGMENTS

The authors would like to thank the CERN Electronics Design Office and Assembly Workshop, Creotech, N.A.T., Struck Innovative Systeme, Javier Serrano and Erik van der Bij for their support during the development and commissioning of the SPS LLRF project.

REFERENCES

[1] G. Hagmann, P. Baudrenghien, J.D. Betz, J. Egli, F.J. Galindo Guarch, G. Kotzian, *et al.*, "The CERN SPS Low Level RF upgrade Project", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 4005–4008, doi:10.18429/JACoW-IPAC2019-THPRB082

[2] White Rabbit project website, https://ohwr.org/project/white-rabbit/wikis

[3] SIS8300KU digitizer, manufacturer webpage, https://www.struck.de/sis8300-ku.html

[4] DESY DS8VM1 up/downconverter RTM module, project webpage, https://techlab.desy.de/products/rtm/drtm_ds8vm1/index_eng.html

[5] AMC FMC Carrier, Zynq Ultrascale variant (AFCZ), project webpage, https://github.com/elhep/AFCZ

[6] M. Lipiński, "Real-Time distribution of magnetic field values using White Rabbit the FIRESTORM project", 2017. https://www.ohwr.org/attachments/5795/BE-CO-TM-WR-BTrain.pdf

[7] NAT MTCA MCH and CPU modules, manufacturer webpage, https://www.nateurope.com/products/NAT-MCH-PHYS.html

[8] MTCA.4 RF Backplane, product webpage, https://techlab.desy.de/products/other_products/rfb/index_eng.html

[9] WR eRTM14/15, project webpage, https://ohwr.org/project/ertm15-llrf-wr

[10] White Rabbit PTP Core (WPRC), project webpage, https://ohwr.org/project/wr-cores/wikis/Wrpc-core

[11] Microchip SyncServer S650 GPS Receiver, product webpage, https://www.microsemi.com/product-directory/gps-instruments/4135-syncserver-s650

[12] WR Switch, project webpage, https://ohwr.org/project/wr-switch-hw/wikis

[13] WRS-LJD Daughterboard, project webpage, https://ohwr.org/project/wrs-low-jitter/wikis

[14] M. Rizzi, M. Lipiński, P. Ferrari, S. Rinaldi, and A. Flammini, "White Rabbit clock synchronization: ultimate limits on close-in phase noise and short-term stability due to FPGA implementation", *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 65, pp. 1726-1737, Sept. 2018. doi:10.1109/TUFFC.2018.2851842

[15] T. Włostowski, "Achieving deterministic phase in Xilinx GTX transceivers ", https://ohwr.org/project/white-rabbit/wikis/deterministic-phase-in-xilinx-gtx-transceivers

[16] H.A. Silva and G.B.M. Bruno, "openMMC: An Open Source Modular Firmware for Board Management", in *Proc. PCaPAC2016*, Campinas, Brazil, Oct. 2016, pp. 94–96. doi:10.18429/JACoW-PCaPAC2016-THPOPRPO04, 2017.

[17] D. Barrientos, J. Molendijk, and G. Hagmann, "A 1 GHz RF Trigger Unit implemented in FPGA logic", https://cds.cern.ch/record/2314652/files/1803.09041.pdf

[18] CERN Open Hardware Licence, project webpage, https://cern-ohl.web.cern.ch

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR03

PROTOTYPE OF WHITE RABBIT BASED BEAM-SYNCHRONOUS TIMING SYSTEMS FOR SHINE*

P.X.Yu, Y.B. Yan†
Shanghai Advanced Research Institute, Chinese Academy of Sciences
201204 Shanghai, P.R. China

G.H. Gong, Y.M. Ye
Tsinghua University
100084 Beijing, P.R. China

J.L. Gu, L. Zhao, Z.Y. Jiang
University of Science and Technology of China
230026 Hefei, P.R. China

Abstract

Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE) is under construction. SHINE requires precise distribution and synchronization of the 1.003086MHz timing signals over a long distance of about 3.1 km. Two prototype systems were developed, both containing three functions: beam-synchronous trigger signal distribution, random-event trigger signal distribution and data exchange between nodes. The frequency of the beam-synchronous trigger signal can be divided according to the accelerator operation mode. Each output pulse can be configured for different fill modes. A prototype system was designed based on a customized clock frequency point (64.197530MHz). Another prototype system was designed based on the standard White Rabbit protocol. The DDS (Direct Digital Synthesis) and D flip-flops (DFFs) are adopted for RF signal transfer and pulse configuration. The details of the timing system design and test results will be reported in this paper.

OVERVIEW

Owing to the wide range of applications of X-rays in the research fields of physics, chemistry and biology, facilities with the ability to generate X-rays were developed continuously in the last century. The free electron laser (FEL) is a novel light source, producing high-brightness X-ray pulses. To achieve high-intensity and ultra-fast short wavelength radiation, several X-ray FEL facilities have been completed or under construction around the world [1].

The first hard X-ray FEL light source in China, the so-called Shanghai HIgh repetition rate XFEL aNd Extreme light facility (SHINE), is under construction. It will utilize a photocathode electron gun combined with the superconducting Linac to produce 8 GeV FEL quality electron beams with 1.003086MHz repetition rate.

* Work supported by Shanghai Municipal Science and Technology Major Project
† yanyingbing@zjlab.org.cn

SHINE timing system is design to provide precise clock pulses (Trigger) for drive laser, LLRF, solid state amplifiers, kicker, beam and optical instruments, etc. It will ensure the electron beam is generated and accelerated to the design energy, to produce the free electron laser, while completing the beam and optical parameters measurement and feedback. The White Rabbit (WR) technology was evaluated and will be adopted.

ARCHITECTURE

SHINE timing system is composed of one master node, WR switchs and more than 500 slave nodes. The master node receives reference signal from the synchronization system. The switches distribute the clock to all the nodes in the network using a hierarchical architecture. The node basic functionality comes in the form of an IP Core called WR PTP Core. They can be standalone trigger fanout modules or FMC boards, which can be embedded in the DBPM and LLRF processor. The system architecture is shown in Figure 1.

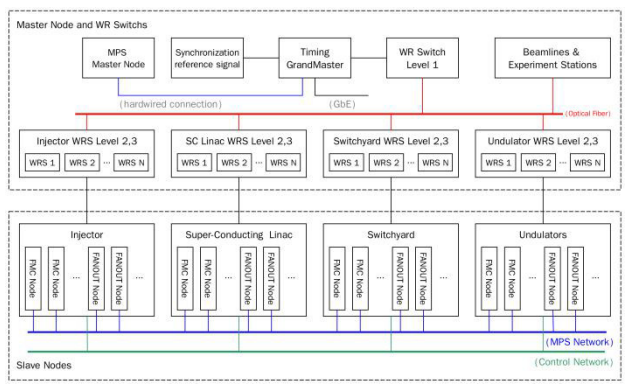


Figure 1: SHINE timing system architecture.

Three functions are designed: beam-synchronous trigger signal distribution, random-event trigger signal distribution and data exchange between nodes. The frequency of the beam-synchronous trigger signal need be divided according to the accelerator operation mode. Each output pulse need be configured for different fill modes.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

SHINE requires precise distribution and synchronization of the 1.003086MHz (1300/1296) timing signals over a long distance of about 3.1 km. The beam-synchronous trigger signal distribution is the basic and priority function.

The standard White Rabbit network operates at 125/62.5MHz clock. If the repetition frequency of SHINE is 1.0MHz, 1300 MHz RF reference signal can be divided to 10MHz as the reference signal. The salve nodes output the trigger signals at the specified time, such as 1us, 2us, 5us, etc. But the repetition frequency is 1.003086MHz, we need to find the new technical routes.

White Rabbit Trigger Distribution

White Rabbit Trigger Distribution (WRTD) is a generic framework for distributing triggers (events) between Nodes over a White Rabbit network [2]. WRTD nodes receive input events and distribute them to other nodes over WR in the form of network messages that are used to transfer the timestamp of the input event. The receiving nodes are programmed to execute some output event (action) upon reception of a particular message, potentially with some fixed delay added to the timestamp [3]. See Fig. 2:

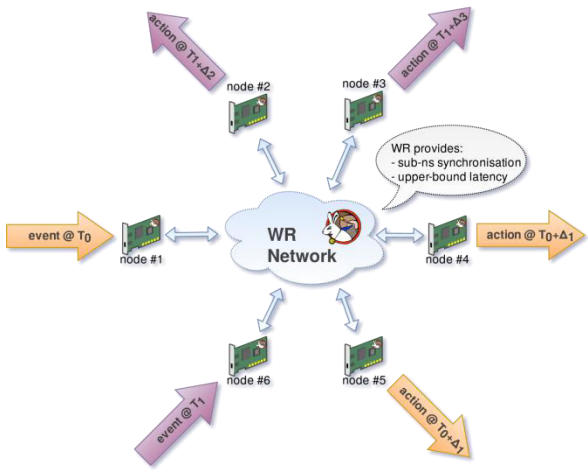


Figure 2: White Rabbit Trigger Distribution[2].

WRTD was used by CERN and SXFEL-UF (Shanghai soft X-ray Free-Electron Laser User Facility). The number of trigger sources in the system and the trigger repetition rate is limited by the network bandwidth and maximum acceptable latency requirements.[4] The SHINE beam-synchronous trigger signal could theoretically be distributed via WRTD, but the repetition rate too high. The 10 Gigabit White Rabbit switches are required, which are no commercial products yet.

The random-event trigger signal distribution is an extension function of SHINE timing system. It is used to distribute various event signals, such as beam loss, machine snapshot, etc. This function is achieved in our two prototypes by WRTD similar technology.

RF over White Rabbit

In White Rabbit network all nodes have the same reference frequency and time. The master node phase locks its DDS (Direct Digital Synthesis) to the RF input, and broadcast the DDS control words including a TAI timestamp. All slave nodes update their DDSes with the received control word at the same moment [4,5]. See Fig. 3:

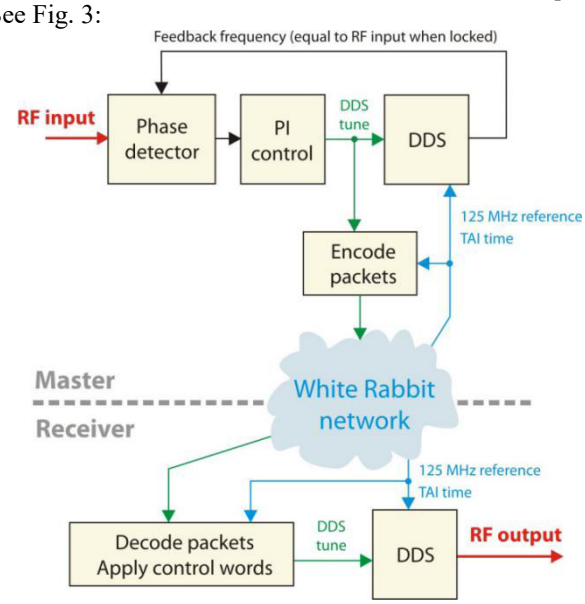


Figure 3: RF over White Rabbit[3].

This technology could be adopted for our beam-synchronous trigger signal distribution, but it needs further development. The slave nodes should output the low-jitter pulse signal. Meanwhile, it is expected that the pulses can be configured for different fill modes. We carried out the prototype (II) development with the state key laboratory of particle detection and electronics, University of Science and Technology of China[6].

Non-Standard Clock Transmission

The repetition frequency of SHINE is 1.0030864MHz. If we build a customized White Rabbit network with non-standard PTP core, we can use the frequency division of 1300MHz as the reference signal instead of GPS/Cesium clock. This was originally just an idea, verified by the prototype (I, II) development. The prototype I was completed in cooperation with with the department of engineering physics, Tsinghua University. This will implement the beam-synchronous timing system with more simpler architecture than RF over White Rabbit.

PROTOTYPE I DEVELOPMENT

The standard White Rabbit network operates at 125/62.5MHz clock, using the external 10 MHz reference signal. The center frequency of VCXO (Voltage-Controlled Crystal Oscillator) is 25MHz. In order to minimize the modifications to the standard White Rabbit equipment, the initial solution was to replace the VCXO of the switches and nodes and shift the operating frequency to 67.708305MHz (1.003086MHz×135/2). The

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR03

system worked using the customized 27.083MHz VCXO, and less than 10ps jitters was achieved.

It is found that the frequency factor 135/2 is not conducive to the phase calculation and pulse delay. Therefore, the operating frequency was changed to 64.197504MHz (1.003086MHz × 64), which is easy to generate 2^N divisions and thus obtain machine clock. Meanwhile, this frequency can be configured from the standard VCXO (25MHz × 52/81), not the customized VCXO.

There is a clear proportional relationship between the standard second and the pseudo-seconds (~0.9969s). The machine clock phase relationship is also determined.

Standard time format:

[seconds : nanoseconds : sub-nanoseconds]

Non-Standard time format:

[pseudo-seconds : clock integer period : phase]

The prototype master node and slave node are shown in figure 4 and 5. The master node receives the random-event trigger signal input. Different functions can be enabled or disabled via the digital signal from MPS (Machine Protection System). The level 1 WR switch is configured as the Grandmaster mode, which receives the external reference and PPS signal.

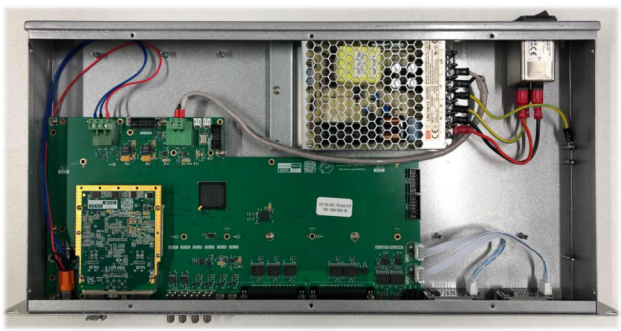


Figure 4: Prototype I master node.

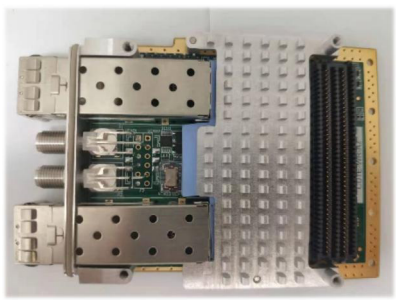


Figure 5: Prototype I salve node.

The slave node is FMC board (FPGA Mezzanine Card), which follows the ANSI/VITA 57.1-2019 standard. There are two SFP ports and two SMA connectors on the front panel. One SFP port is used to connect to the White Rabbit network and the other is in reserve. It will be used to connect to the standard network for status monitoring in the future. One SMA connector outputs the beam-synchronous trigger signal, the other can be configured as interlock input. More signal outputs are developed through the FMC connector, including 8 independent beam-synchronous trigger signal channels with adjustable delay and pulse width, 4 independent random-event trigger signal channels.

PROTOTYPE II DEVELOPMENT

Prototype II implements both the two technical routes: the non-standard clock transmission and RF over White Rabbit. The details of the latter are described below.

The system architecture is shown in Figure 6. Based on the standard White Rabbit network, the master node converted the 9.027778MHz (1.003086MHz × 9) RF signal into the frequency and phase information. Then the data is transmitted to all slave nodes. The slave nodes use the received date to recover the clock and realize precise distribution and synchronization of the machine clock.

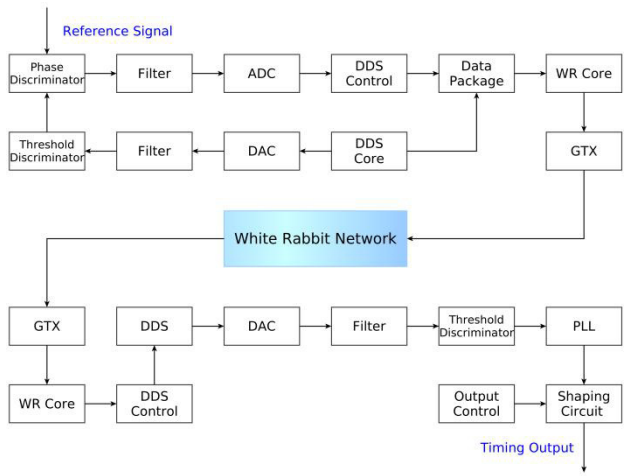


Figure 6: Prototype II system architecture.

The prototype master node and slave node are shown in figure 7 and 8. The master node is configured as the Grandmaster mode, which receives the external reference signal and random-event trigger signal. The standard WR switches are adopt to connect the slave nodes.

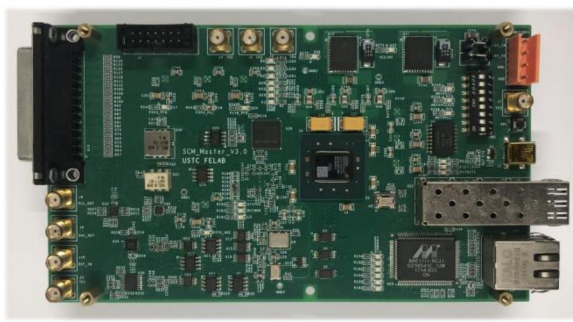


Figure 7: Prototype II master node.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

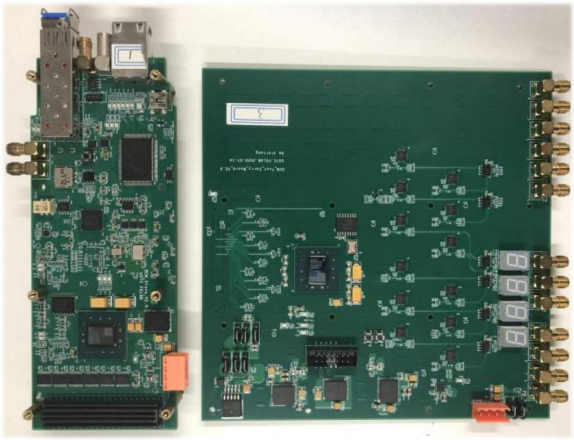


Figure 8: Prototype II salve node.

The slave node is a non-standard FMC card. It is used to verify multiple functions, including signal recovery, pulse output, frequency division, delay adjustment and so on. The diagram of fan-out and shaping circuit is shown in Figure 9.

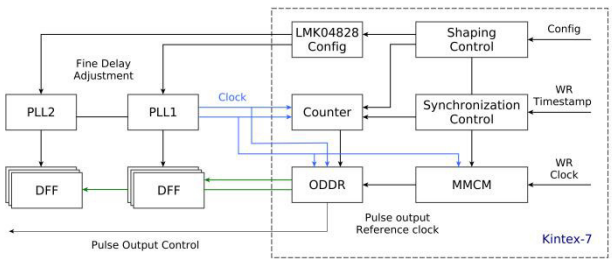


Figure 9: Fan-out and shaping circuit.

The circuit is mainly composed of the FPGA, off-chip PLL and DFF. The FPGA is responsible for frequency division, pulse width and coarse delay adjustment. The off-chip PLL is used for the signal fan-out and fine delay adjustment. The DFF latches the output of FPGA to ensure the quality and accuracy of the final pulse output.

TEST RESULTS

A series of tests were carried out to confirm the performance of the prototype. The following parameters are verified for beam-synchronous trigger and random-event trigger signal distribution.

- Jitter between the output of slave node and the external reference signal.
- Jitter between the outputs from different slave nodes.
- Skews between two slave nodes after powering up and down.
- Delay and pulse width adjustment.
- Temperature drift.

Prototype I Test

The test platform of Prototype I is shown in Figure 10 and 11. The GPS/Rubidium clock provides a stable 10MHz external reference signal for the arbitrary waveform generator, not for White Rabbit equipment. One channel of periodic square wave signal (1.003086MHz×16) is used as the non-standard external reference signal. The other channel is used as PPS signal after frequency division by DG645. The level 1 WR switch is configured as the Grandmaster mode. The computer is connected to the non-standard WR switch via the conversion module, which is used to set the parameters of the WR equipment. The Keysight MSOS604A Oscilloscope is used to measure the output of slave node and the external reference signal.

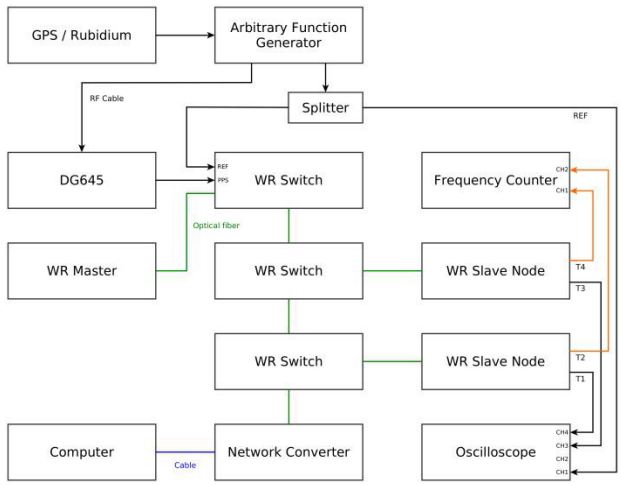


Figure 10: Prototype I test platform diagram.

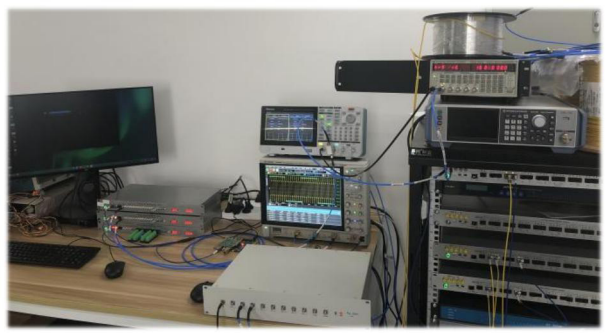


Figure 11: Prototype I test platform.

For the beam-synchronous trigger signal distribution, the jitter between the output of slave node and the external reference signal is less than 10ps. The jitter between the outputs from different slave nodes is less than 5ps. The temperature drift test results are shown in the figure 12. For the random-event trigger signal distribution, the jitter between the output of slave node and the external reference signal is less than 60ps.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBR03

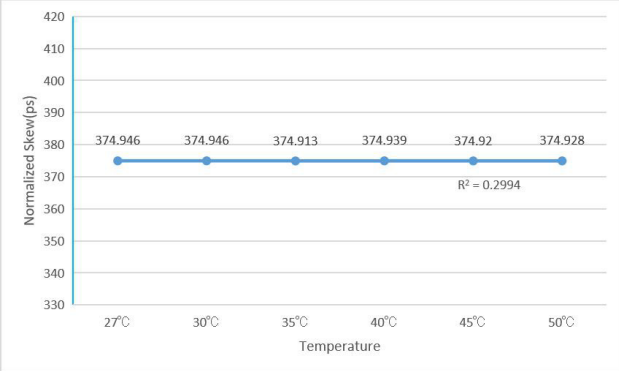


Figure 12: Result of the temperature drift test.

Prototype II Test

The test platform of Prototype II is shown in Figure 13. The GPS/Rubidium clock provides a stable 10MHz external reference signal for the arbitrary waveform generator and WR switch. The master node is configured as the Grandmaster mode, which receives the RF signal (1.003086MHz×9). The computer is connected to the WR equipment via the standard network switch. The Keysight MSOS604A Oscilloscope is used to measure the output of slave node and the external reference signal.

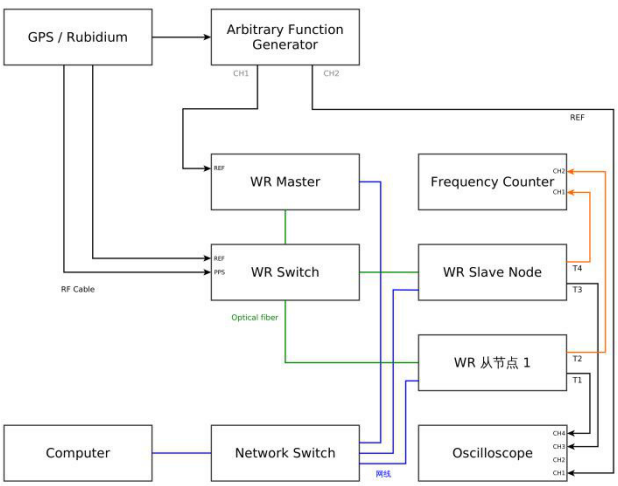


Figure 13: Prototype II test platform diagram.

For the beam-synchronous trigger signal distribution, the jitter between the output of slave node and the external reference signal is less than 20ps. The jitter between the outputs from different slave nodes is less than 10ps. For the random-event trigger signal distribution, the jitter between the output of slave node and the external reference signal is less than 35ps.

CONCLUSION

Two prototype systems were developed, both containing three functions: beam-synchronous trigger signal distribution, random-event trigger signal distribution and data exchange between nodes. The non-standard clock transmission was proposed and verified. The prototype development has just been completed and we will do further test and evaluation for the SHINE project.

ACKNOWLEDGMENT

We would like to thank CERN colleagues of the Hardware and Timing Section, Beam Controls Group (BE-CO) for the discussions and suggestions.

REFERENCES

[1] K. Li and H.X. Deng, "Systematic design and three-dimensional simulation of X-ray FEL oscillator for Shanghai Coherent Light Facility", Nuclear Instruments & Methods in Physics Research, A, vol. 895, pp. 40-47, 2018. doi:10.1016/j.nima.2018.03.072

[2] White Rabbit Trigger Distribution, https://ohwr.org/project/wrtd/wikis/home.

[3] *White Rabbit Trigger Distribution Documentation*, Dimitris Lampridis, Oct 01, 2019, https://readthedocs.org/projects/wrtd/downloads/pdf/v1.0.0/

[4] T. Włostowski et al., "Trigger and RF Distribution using White Rabbit", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 619-623. doi:10.18429/JACoW-ICALEPCS2015-WEC3O01

[5] Distributed RF over White Rabbit, https://ohwr.org/project/wr-drf/wikis

[6] Jinliang Gu, Lei Zhao *et al.*, "Prototype of Clock and Timing Distribution and Synchronization Electronics for SHINE", *IEEE Transactions on Nuclear Science*, vol. 68, pp. 2113-2120, 2021. doi:10.1109/TNS.2021.3094546

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV001

SUPERVISORY SYSTEM FOR THE SIRIUS SCIENTIFIC FACILITIES *

L.C. Arruda†, G.T. Barreto, H.F. Canova, J.V. B. Franca, M.P. Calcanha,
Brazilian Synchrotron Light Laboratory (LNLS) Campinas, Brazil

Abstract

A general supervisory system for the scientific facilities is under development at Sirius, the Brazilian 4th generation synchrotron light source. The data generated by different classes of equipment are generally available via EPICS or industrial protocols such as OPC-UA provided by commercial automation systems. However, as the number of beamlines and laboratories expands, the effort to properly gather, display and manage this data also scales up. For this reason, an aggregating supervisory system is proposed to monitor the systems: power distribution, personal safety, beamline components, cryogenic fluids; mechanical utilities, air conditioning, among others. This work presents the overall system architecture, functionalities, and some user interfaces.

INTRODUCTION

The general supervisory is a Supervisory Control and Data Acquisition (SCADA) system. It aims to provide a simplified web visualization and concentrate the creation of alarms notifications of Sirius' scientific facilities. The system final users are the facilities' support groups.

This article is divided into parts that discuss architecture, information organization, graphical user interface (GUI), and key performance indicators (KPIs).

ARCHITECTURE

The parts that compose the system and the communication protocols used appear in Fig 1. The main server is the Siemens WinCC Unified Runtime [1] and it aims to exchange data with other devices, process information, handle alarms and communicate with the web clients. The Experimental and Industrial Control System (EPICS) [2] data server is used for concentrating the Epics Process Variables (PVs) data from the installations' EPICS servers and provide data to the main server using only one OPC Unified Architecture (OPC UA) [3] connection. The external communication server is intended to expand the way notifications can reach the user, in addition to the web clients alarms tables, providing services like sending emails, Telegram [4] and SMS messages. The SMS messages are sent by the Siemens 3G modem Scalance M874-3 [5] using a Socket protocol between the modem and the external communication server. The connection between the programmable logic controllers (PLCs) and the Main server uses a Siemens proprietary protocol based on TCP/IP, in these set of PLCs are included the equipment and personal protection systems [6].

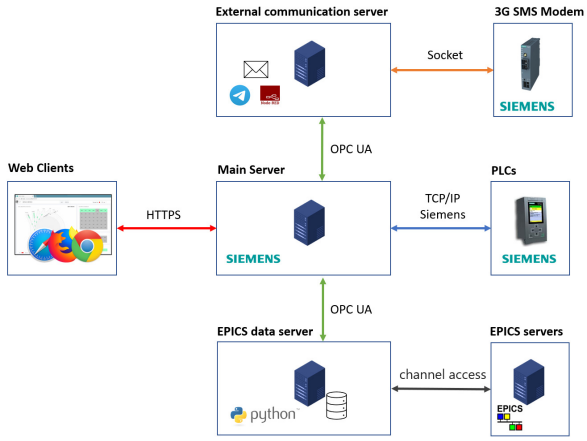


Figure 1: Architecture and communication schema.

Some of the Main server performance characteristics include the capacity to connect with 600,000 external tags, to have 200,000 internal tags, generate 200,000 discrete alarms, connect to 128 Siemens S7 SIMATIC PLCs, 10 OPC UA clients, and 100 web clients [7].

Access to the SCADA system uses password-protected authentication. The system is available only in the internal network, and user credentials needed are the same used to access other computational resources.

A software was developed to run on the EPICS data server using PyEPICS [8]. The update of data obtained from installation EPICs servers is configurable and is set to occur every second, this period is seen appropriated due to the absence of the need for quick diagnoses. Right after the PV is read, the OPC UA variable is updated.

To record trends and to perform more specifics diagnostics, when a correlation is needed, the tool used is the Archiver [9].

INFORMATION ORGANIZATION

Usually, the users are interested in specific parts of the scientific facilities related to their expertise area such as electrical power distribution, mechanical utilities, vacuum, radiological protection, among others. Considering that support groups' work structure may be changed, the supervisory information was organized by the facilities subsystems and the user is responsible for finding their interest area.

The GUI hierarchy pattern is shown in a general form in Fig. 2. The screens are currently grouped in the general view, infrastructure, beamlines, safety, statistics, and support labs. Inside each group, there is a division by subsystem and inside each subsystem, there are one or more personalized screens related to the subsystem and one

* Work supported by the Brazilian Ministry of Science, Technology and Innovation
† lucas.arruda@lnls.br

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV001

screen used to view and configure the alarms and other notifications.

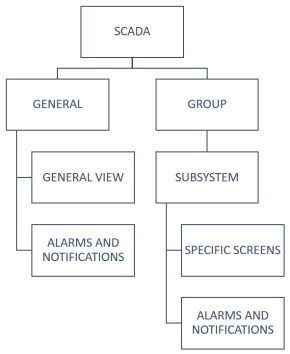


Figure 2: GUI hierarchy pattern.

Alarms and notifications screens by subsystem were designed to allow the group of people responsible for each subsystem to manage the configuration and acknowledgment of alarms and notifications.

GUI

The Sirius' general screen, radiological protection, electrical power distribution, and alarms are currently in beta version or in development and are presented to show some of the GUI possibilities. In addition, the screens for components, vacuum, utilities, HVAC (heating, ventilation, and air conditioning), cryogenic systems among others are predicted.

Sirius General

The Sirius' general screen plans to provide a simplified view of the hutches' status, beamlines, and subsystems. This screen is in Fig. 3.

In this screen, the hutches' status can indicate: Fail (Beam is not allowed in hutch), beam on (allowed beam on the hutch), and power off (a failure related to electrical power distribution has occurred). On the right, two matrices show the status of the beamlines, subsystems, and beamlines' gamma shutter authorization to open. Besides providing a simplified diagnostic, the existence of these matrices facilitates the view if the user uses a smartphone.

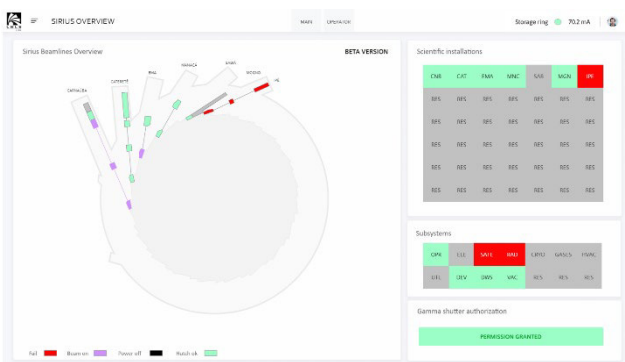


Figure 3: The Sirius' general screen showing the status of beamlines, hutches, subsystems, and authorization to open front-ends' gamma shutters.

Radiological Protection

The radiological protection screen follows the same principle used in Sirius general screen, the main difference is that the status change to the radiological protection context. The screen is in Fig. 4.

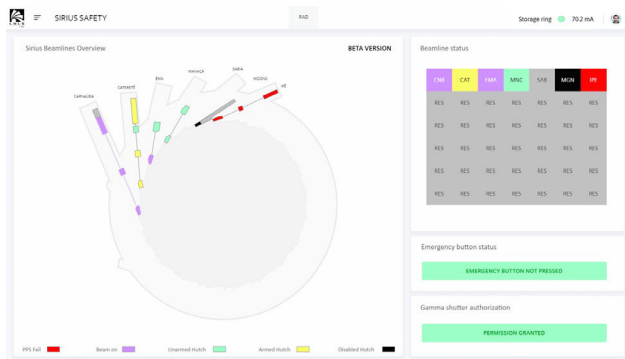


Figure 4: The radiological protection screen showing the status of beamlines, hutches, emergency buttons, and authorization to open de front-ends' gamma shutters.

Power Distribution

The screens related to electrical power distribution are divided into two parts: A general view and the beamlines screen. The General view is in Fig. 5 and an example of the beamlines screens is shown in Fig. 6.

The general view indicates the status of beamlines, the building power distribution panels, emergency buttons, and power consumption. The overview part shows the power being consumed by the 220 VAC, 380 VAC, and 220 VAC provided by the uninterruptible power supply (UPS). The Sirius mini-map indicates the hutches' status and the power distribution panels indicating their locals in the installation.

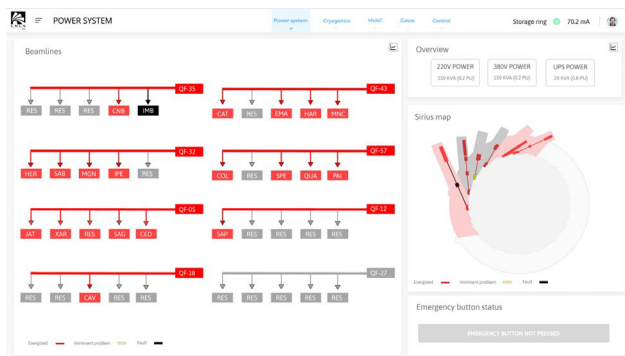


Figure 5: The electrical power distribution general screen showing the status of beamlines, hutches, emergency buttons, and building power panels.

The beamlines screens have a unifilar diagram that indicates the energized and de-energized regions, currents and allows to select a device to view more detailed information about the device in "Component data". In "Overview" is available the total amount of power being consumed in real time by the alternated current circuits. In "UPS" region is shown the status of the central 220 VAC UPS and the 24

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV001

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

VDC UPS that feed the emergency light and other emergency circuits related to reset the safety relays for example.

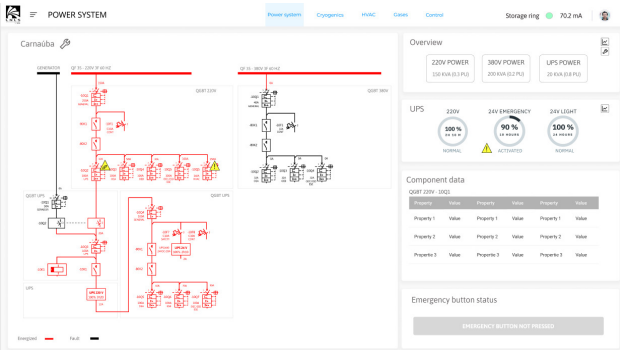


Figure 6: The beamline electrical power distribution screen showing the unifilar diagram, power consumption and status of UPS and emergency buttons.

If there's any condition that requires the user attention, a warning symbol is shown to direct then.

Notifications

The notification screen is divided into two parts: The "Attention required", which manages the alarms and allows to configure notifications settings, and the "Log" which shows the recorded data of all events of interest related to the subsystem. The notifications screen is in Fig. 7.

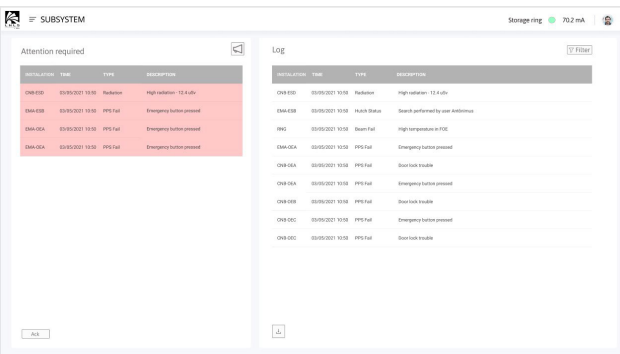


Figure 7: The subsystems' notification screens indicating the alarms that require user attention on the left and the log data on the right.

In "Attention required" exists a button on the lower part of the screen that is used to acknowledge the events and clear the stack. Using a button in the superior part of the screen the user can upload a configuration file that contains the notifications settings that can only be made by authorized users. All alarms configured to be shown in the "Attention required" region generate notifications.

In "Log" there is a button on the superior part of the screen that can be used to filter the data shown in the log table, the main filter parameters are beamline and the period. Using the button on the lower part of the screen, the user can download a file that contains the filtered log data.

KPI

The KPIs aim to measure performance. These indicators are very dependent on the activity type and are used in the decision take process. In the supervisory system, the KPIs are planned to be implemented in the future.

In the beamline's context, the main parameter of interest is the beam availability to experiments during the programmed beamline operation period while the subsystems has nominal conditions. If the availability in these conditions is not satisfied, there's also interested to know which subsystem was not available, the local of the problem, and the period. These measurements allow directing the maintenance and the improvements by the support groups.

In addition to the availability, other parameters are being considered for beamlines and subsystems are the Meantime Before Failure (MTBF) and the Mean Time To Repair (MTTR).

CONCLUSIONS

The SCADA system is complex by involving a high number of signals provided by different sources, to demand detailed knowledge of the facilities and by considering the user's interest from different areas. The possibility of a high number of connections, simplified GUI visualization, performance indicators, and the web access present in this work are key points to achieve the system's requirements.

ACKNOWLEDGMENTS

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, and Innovation and the contribution of the LNLS team involved in the development of this supervisory system.

REFERENCES

[1] Siemens Wincc Unified,https://new.siemens.com/global/en/products/automation/simatic-hmi/wincc-unified.html

[2] Experimental Physics and Industrial Control System, https://epics.anl.gov/

[3] OPC Foundation, https://opcfoundation.org/about/opc-tech-nologies/opc-ua/

[4] Telegram, https://telegram.org/

[5] Scalance M874-3, https://support.industry.siemens.com/cs/pd/345192?pdti=pi&dl=en&lc=en-DE

[6] L. C. Arruda et al., "Equipment and Personal Protection Systems for the Sirius Beamlines", presented at *the 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, Shanghai, China, Oct. 2021, paper WEPV034.

[7] WinCC Engineering V16 – Runtime Unified Performance features, https://support.industry.siemens.com/cs/docu-ment/109773780/simatic-wincc-wincc-engineering-v16-runtime-unified-?dti=0&lc=en-RO

[8] PyEpics: Epics Channel Access for Pyhton, https://github.com/pyepics/pyepics

[9] Experimental Physics and Industrial Control System, https://epics.anl.gov/extensions/ar/index.php

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV004

OPEN-HARDWARE KNOB SYSTEM FOR ACCELERATION CONTROL OPERATIONS

E. Munaron, M. Montis, L. Pranovi, INFN-LNL, Legnaro, Italy

Abstract

Nowadays technologies in LINAc facilities brought the common Human-Machine Interfaces (HMIs) to be more aligned to the standards coming from the information technology (IT) and the operators started to interact to the apparatus with the common computers' instruments: mouse and keyboard. This approach has both pros and cons. In order to minimize the cons and with the idea of providing an alternative to interact with HMIs, we tried to design and realize an open-hardware knob system solution.

INTRODUCTION

In the market, in addition to the standard devices (such as keyboard, mouse, and so on), there are several HMI controllers used to interact with personal computers. This kind of hardware is based on encoders and buttons and it needs dedicated drivers for specific operating systems (OSs) or software to work.

Sometimes software tools and applications adopted by the control system environment can introduce limitations and constraints in the usage of these commercial devices.

To overcome this limitation (in terms of drivers and OS interface), the need for a new technical solution would be suitable.

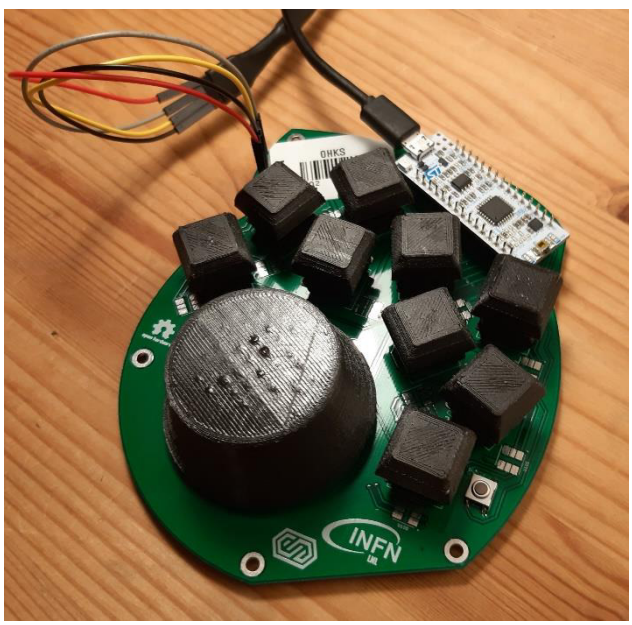


Figure 1: The open-hardware knob controller prototype.

THE OPEN-HARDWARE KNOB CONTROLLER

Because of the heterogeneous situation in terms of hardware and software solutions adopted for the different functional systems composing the facilities at INFN National Laboratories in Legnaro, the new controller should guarantee to be easily interfaced with several kinds of instruments and devices. According to this assumption and based on the talk by Kerry Scharfglass during KiCon 2019 [1], we started to study and design a new knob controller equipped with an encoder and buttons which has to achieve the following goals:

- Easy to configure
- Multi-platform
- Cost-optimized
- Adopt public licenses for both hardware and software

Unlike many proprietary solutions, this hardware interface can be recognized as a USB keyboard from any kind of device and is virtually compatible with any modern operating system. Key buttons and encoder can be programmable and dynamically reconfigurable by the user if needed.

In addition to the technical assumptions previously mentioned, we've chosen to release the project under Open Hardware License (OHL) [2] and GNU General Public License version 3 (GPLv3), so any developer or user interested in this work will be free to reproduce, develop, customize and share his product.

ELECTRONICS

The hardware (Figure 1) has been structured on a Printed Circuit Board (PCB) where the following parts are mounted:

- 9 Cherry-MX type keys
- 1 mechanical encoder
- 1 microcontroller

The keys are used to simulate, at the firmware level, the pressure of one keyboard button or a combination of them. Each key is equipped with anti-bounce circuitry and adequate ESD protection (Human-Body Model - HBM) to prevent a malfunction or breakdown of the electronics.

The microcontroller is devoted to managing the USB HID communication to the target system (such as a personal computer or a control device). To have multiple options in terms of hardware solutions for the microcontroller, the PCB has been realized to be compatible with both ARM STM32 microcontroller [3] and Teensy LC module [4]. This degree of freedom is very useful because it let developers select the preferred software toolchain for firmware implementation: for example, the Teensy LC module

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV004

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

is compatible with Arduino Integrated Development Environment (IDE) [5], the famous opensource solution used to write and upload programs to Arduino compatible boards or, with the help of third-party cores, other vendor development boards.

Following the open-source philosophy adopted for the entire project, the PCB schema has been built using KiCad open-source software [6].

MECHANICAL COMPONENTS

Under the mechanic aspect, the main goal the project wanted to achieve was to provide a complete system buildable from scratch, minimizing the cost in terms of components. For this reason, different parts have been designed to be realized through 3D printers and Fused Deposition Modeling (FDM) technology. Among the different 3D printing solutions, FDM technology has been considered the best choice because it is suitable for creating robust, durable, and dimensionally stable components, with the highest degree of accuracy and repeatability with low costs. As a matter of fact, 3D printers are quite diffused in several professional heterogeneous environments and among makers because of the commercial costs for devices, tools and raw materials.

Based on this choice, key covers and encored knob have been designed to be produced using FDM technology as visible in Figure 2, while the entire device cover has not been finalized due to the fact that additional ergonomics studies on the entire controller are ongoing.

The entire set of 3D models have been done with FreeCAD [7] (Figure 3), according to the philosophy adopted by the project.

Figure 2: Mechanical parts realized with FDM technology.

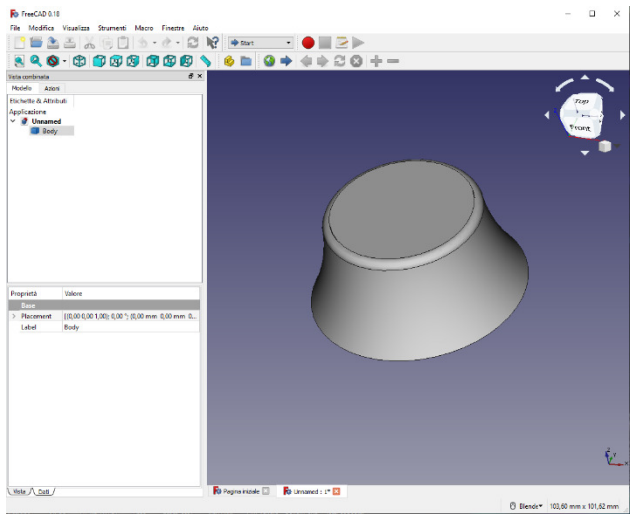


Figure 3: FreeCAD knob schematics.

FIRMWARE

Based on the microcontroller used, the preliminary firmware version has been defined using Arduino IDE and Teensyduino add-on [8] as software toolchain (see Figure 4).

This simple application is composed of the main Arduino sketches file (Arduino program language based on C/C++) and an external C header file. In particular:

- The main sketches file is devoted to establish the serial port communication and manage the events;
- The C library defines the maps for the key buttons and the knob controller. In this place it is possible, for example, to customize the response in case of buttons pressed combinations;
- Additional standard Arduino libraries.

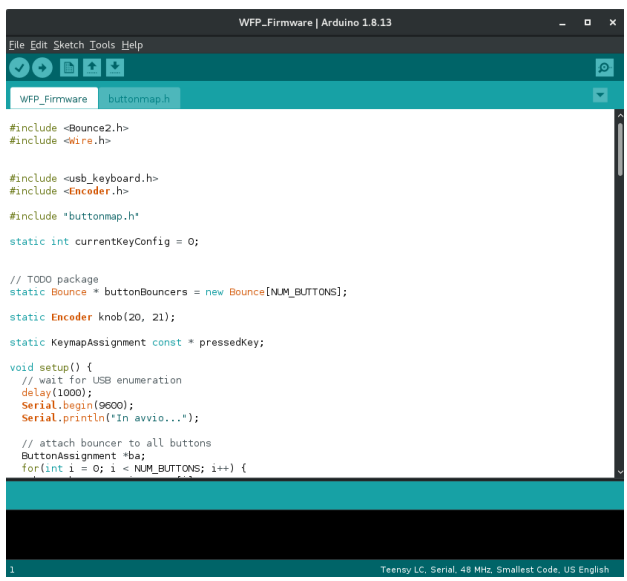


Figure 4: Firmware code realized with Arduino IDE and Teensyduino packages.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV004

FIRST TESTS

Preliminary tests on different operative systems let us verify the correct functionality in terms of electronics and firmware: in every setup, the device has been automatically recognized as a USB keyboard. Trying to use a heterogeneous set of applications (word processors, draw applications, etc.), the device substituted the normal keyboard without any issue (according to the buttons map pre-loaded in the Firmware).

FURTHER DEVELOPMENTS

After the preliminary R&D performed and the test results obtained, further developments will be focused on two different areas of interest:

- Optimize the electronics and the mechanical parts to simplify schematics for PCB production and 3D printing processes
- Further investigations in ergonomics of the device, taking into account the feedback coming back from the beta testers, to realize a comfortable controller for end-users experience

In addition to that, a long-term goal the project wants to achieve is an advanced integration with the programs and software used for the control system; as consequence, deeper studies and tests with control system HMI applications will be performed, in order to adopt this hardware solution as a standard for INFN-LNL linear accelerators facility.

CONCLUSION

The need for a simpler and more intuitive solution for beam transport and experimental control has been the starting point to design and develop a new controller for linear accelerators.

The prototype realized is a multi-platform, easy to configure solution where most of the costs have been minimized through the usage of fast prototyping technologies, such as 3D printing. In addition to that, the product follows the open-source philosophy, and all the design parts are freely available under OHL and GPLv3 licenses.

First feedbacks coming from the test are very promising and let us confirm the technology assumptions made during the design stage; at the same time, further development will be done to optimize different aspects (electronics, mechanics, performances) and realize a more efficient and comfortable control device.

REFERENCES

[1] Kerry Scharfglass "Alternative User Interfaces for KiCad" KiCon 2019, https://2019.kicad-kicon.com/talks/

[2] OHL, https://ohwr.org/cernohl

[3] STM32 microcontrollers, https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

[4] Teensy LC, https://www.pjrc.com/teensy/teensyLC.html

[5] Arduino IDE, https://www.arduino.cc/en/software

[6] KiCad EDA, https://www.kicad.org/

[7] *FreeCAD,* https://www.freecadweb.org/

[8] Teensyduino package, https://www.pjrc.com/teensy/teensyduino.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

VIRTUAL REALITY AND CONTROL SYSTEMS: HOW A 3D SYSTEM LOOKS LIKE

L. Pranovi, M. Montis, INFN-LNL, Legnaro, Italy

Abstract

Virtual Reality (VR) technology and its derivatives are mature enough to be used in environments like a nuclear research laboratory, to provide useful tools and procedures to optimize the tasks of developers and operators. Preliminary tests were performed [1] to verify the feasibility of this technology applied to a nuclear physics laboratory with promising results. Since this technology is rapidly diffusing in several different professional heterogeneous environments, such as medicine, architecture, the military and industry, we tried to evaluate the impact coming from a new kind of Human-Machine Interface based on VR.

PRELIMINARY WORKS

In a complex environment like a nuclear facility, many tasks can be difficult to execute because of the limitations (in terms of time and availability) due to the normal operations. In this scenario, the usage of Virtual Reality Technology can be an extraordinary way to overcome these limitations.

Among the several possibilities offered by the daily work, we focused on three main aspects: data collection (used to verify the incoherencies among the data provided by the groups involved in specific tasks and projects and to correct them on design and documentation), training (used to train operators to work in parts of the particle accelerator, giving them the opportunity to familiarize with the system despite its real availability) and machine maintenance.

The studies executed and the proof of concept designed and implemented verified the maturity and the versatility of this technology: the application developed gave us preliminary good feedbacks for the areas of interest previously mentioned and the results were very promising and pushed us to extend studies and application functionalities of the prototype, embracing different hardware solutions and integrating heterogeneous data and information.

At the same time, the work done has been consolidated and extended: the training based on VR technology has obtained several good feedbacks (Figure 1) and, accordingly, a new VR experience for radioprotection staff is under discussion.

VIRTUAL REALITY AND AUGMENTED REALITY TECHNOLOGIES

In the last decade, the mayor IT companies invested a lot of effort in VR technology and, as results, several products arrived on the market.

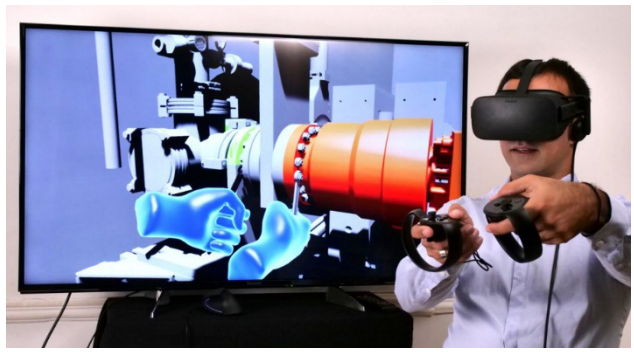


Figure 1: Beta tester for VR training.

At the same time the cost related to this kind of products become more attractive for the end user. In these last years, while VR devices are becoming quite common, first AR (Augmented Reality) type controllers are going to be available on the market (with high costs) [2].

These two kinds of technologies have different characteristics and, as consequence, offer different experiences to the user. Comparing them, it is possible to analyse VR in an interactive computer-generated experience taking place within a simulated environment, that incorporates mainly auditory and visual, but also other types of sensory feedback. It is rapidly diffusing among several different professional environments, such as medical, architecture, military and industry, with different level of interactions, based on the experience required. On the other hand, AR technology is defined as “an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory and haptic, somatosensory and olfactory. AR can be defined as a system that incorporates three basic features:

- a combination of real and virtual worlds
- real-time interaction
- accurate 3D registration of virtual and real objects.

The overlaid sensory information can be constructive (i.e., additive to the natural environment), or destructive (i.e., masking of the natural environment). This experience is seamlessly interwoven with the physical world such that it is perceived as an immersive aspect of the real environment” [3]. It is possible to say that augmented reality alters one's ongoing perception of a real-world environment, whereas virtual reality completely replaces the user's real-world environment with a simulated one.

However, AR technology has a distinct disadvantage compared with virtual reality: visual immersion. While VR completely covers and replaces your field of vision, AR

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV005

can only project images in a limited area in front of your eyes. This aspect must be kept into account when AR applications are going to be developed for supervision and control.

Figure 2: Panels in AR added as additional virtual desktop in the main control room.

AR IN NUCLEAR FACILITIES

The technology is designed for free movement, while projecting images over whatever you look at, with the possibility of introducing additional information.

For our preliminary study, HoloLens 2 display by Microsoft® [4] were used as AR device: it is a combination of waveguides and light projectors which are inside the enclosure above the brow. HoloLens 2 uses laser light to illuminate the display [5].

HoloLens 2 has been designed as a tool useful in an heterogenous set of environments and operations, such as manufacturing shop floors, field services, construction sites, remote work, sales, teaching and so on. In a facility like a nuclear plant, the introduction of this kind of device can increase performances and help operators in numerous tasks:

- provide an easy way to recall information during apparatus maintenance
- communicate with colleagues and teams through dedicated communication apps
- introduce a new concept of Human-Machine Interface (HMI) where the classical control panels are extended with virtual ones

The principal user case has been the definition of a virtual console used by operators working with the cyclotron apparatus at LNL. A renewal process is ongoing, in parallel with normal operations, in order to upgrade the apparatus high-level control interface. In this scenario, additional screens are reproduced in AR next to the normal control monitors (Figure 2). This configuration lets users have the main screens devoted to monitor and supervise the machine provided in AR while the normal computer-based HMI can be used to interact with the control (sending command and execute procedures).

The AR screens are a reproduction of the console panels, optimized for HoloLens device. This approach simplifies the machine conduction because all the graphical user interfaces are harmonized among them.

Under technical aspect, Unreal Engine [6] software has been used. The choice of this tool has been based on two factors:

- the experience acquired with the work done with VR experiences
- the availability of dedicated Software Development Kit (SDK) optimized for AR technology.

It is important to underline that all the screens defined in AR are devoted only for monitoring: no commands or machine's procedure can be executed via AR technology. This condition has been adopted to guarantee safety and reliability for users and apparatus.

Further studies and tests are ongoing to provide control information during maintenance: the goal we want to achieve is to provide at least the same control panels available to operators working close to the machine during local supervision. Preliminary tests have been done (Figure 3) with promising results (i.e., touchless controls maximize the cleaning of environment and tools), but more effort must be invested in this task.

Figure 3: Preliminary tests with AR technology for maintenance operations.

FURTHER DEVELOPMENTS

The work done and exposed is a preliminary test for a more stable solution of virtual monitoring and it requires further development regarding reliability and usability. The main gaol we want to achieve is to extend the usage of virtual desktops not only for supervision but also for control: this step can have an important impact in term of space optimization and portability.

Concerning this goal, additional studies and investigations are required to evaluate the effort required to realize multi-user and multi-platform solutions.

As the number of AR tools is on the rise, it is also critical to evaluate and select additional tools and software needed to develop more AR experiences.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONCLUSION

Augmented Reality technology is rapidly becoming an important asset in many professional fields. In an environment like nuclear facilities, it has the potential to redesign the concept of Human-Machine Interface and it can extend the usage of control parameters outside the classical control room. The preliminary tests done realizing new control panels in AR confirmed the huge potential of the tool, but further effort is required to fully integrate this technology in an heterogenous system (in terms of software and applications) like the one nuclear facility represents.

REFERENCES

[1] L. Pranovi and M. Montis, "VR as a Service: Use of Virtual Reality in a Nuclear Accelerator Facility", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA101.

[2] "*The Lengthy History of Augmented Reality*". Huffington Post. 15 May 2016.

[3] Wikipedia Definition, *https://en.wikipedia.org/wiki/Augmented_reality*

[4] HoloLens 2 - About Hololens, *https://docs.microsoft.com/en-us/hololens/hololens2-hardware*

[5] HoloLens 2 - Display, *https://docs.microsoft.com/en-us/hololens/hololens2-display*

[6] Unreal Engine, *https://www.unrealengine.com/en-US/*

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV006

DESIGN OF REAL-TIME ALARM SYSTEM FOR CAFe

N.Xie,Y.H.Gua, R.Wang, B.J.Wang, IMP, LAN Zhou 730000, P.R. China

Abstract

In accelerator control, the alarm system is a very important real-time monitoring and control system. In order to find specific failures of accelerator-related equipment in time, improve the high availability of the equipment, and ensure the long-term operation of the accelerator. An accelerator alarm system based on Kafka was designed and built on the CAFe. The system uses Phoebus for architecture deployment. Kafka is used as the streaming platform of the alarm system, which effectively improves the throughput of the system and realizes real-time alarms.In order to realize the function of remote monitoring of data in the central control room, CS-Studio is used to draw the opi interface to deploy to the enterprise WeChat platform to realize remote data monitoring. This system greatly improves the response speed of fault handling and saves a lot of valuable time for accelerator fault handling.

INTRODUCTION

China Initiative Accelerator Driven System (CiADS) plays an important role in the safety of spent fuel handling. This is a global challenge that has not yet been resolved by our country and the international nuclear energy community. Chinese ADS Front-end Demo Linac (CAFe) as a CiADS prototype. Figure 1 is the layout of the CAFe superconducting linear accelerator, which consists of nine parts. CAFe as a prototype of CiADS.Its research purpose is to develop clean, efficient and safe nuclear fission energy, and to solve the future energy supply. Therefore, the stable operation of the CAFe equipment is particularly important for the debugging and stable operation of the beam experiment.

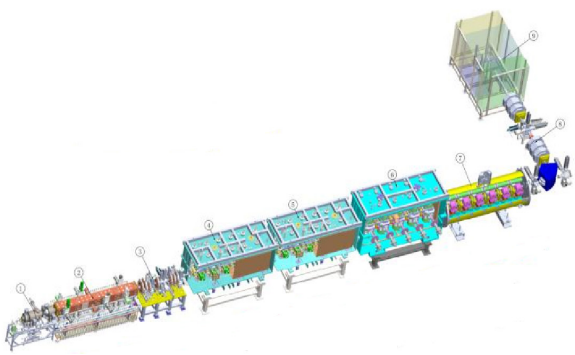


Figure 1: Layout of CAFe Superconducting Linear Accelerator.

SYSTEM STRUCTURE

Figure 2 is the overall framework of the alarm system, which is mainly composed of the control layer, the alarm service layer, the Kafka layer and the application layer.

Control Layer

IOC (input output controller) is the executor of control tasks[1]. It is used to obtain the data of the monitoring equipment and define the alarm threshold of the equipment. The sending of alarm information is based on the setting of the threshold. Each record in the IOC has multiple fields, “SEVR”, “ACKT” “STAT” and “ACKS” are related fields of the alarm and can be customized.

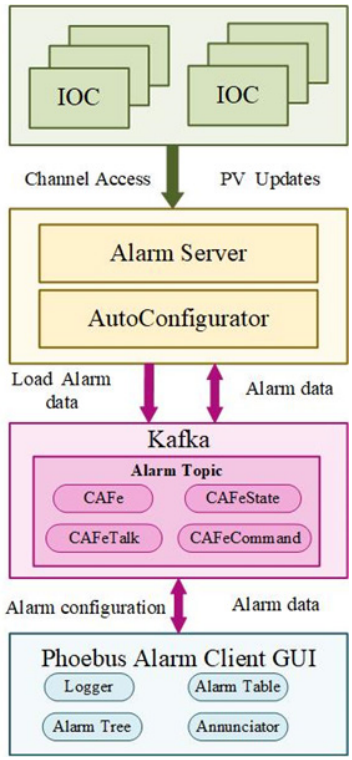


Figure 2: Framework diagram of an alarm system based on Kafka.

Alarm Service Layer

Alarm Server connects all PVs and obtains the records that need to be monitored through the CA protocol and configuration files. At the same time, Alarm Server collects the command information from the upper application “Acknowledge” in the topic "CAFeCommand" in Kafka to confirm the alarm. When the alarm status changes in the records, the Alarm Server will generate a new alarm to update the content in the "AcceleratorState". AutoConfigurator obtains the recorded configuration information and delivers alarm configuration information to the topic “CAFe”.

Kafka Layer

The Kafka layer generates 4 themes: Accelerator,AcceleratorState, AcceleratorCommand, and AcceleratorTalk. Alarm Server obtains current alarm configuration information, real-time alarm status, Acknowledge commands, and voice alarm records from the above 4 topics respectively.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Application Layer

Phoebus subscribes to Kafka Topic to realize visualization of alarm information. It can display Alarm Tree, Alarm Table, Alarm Area Panel, Logger and Annunciator user interface. The staff in the central control room can know whether the accelerator is operating normally according to the alarm interface.

Phoebus

Phoebus is a new version of Control System Studio, which eliminates the dependence on Eclipse RCP and SWT, which is conducive to the development of control system users. Phoebus is the graphical interface of EPICS control system. It provides a simple and easy-to-operate alarm interface for the central control room. The alarm module components are divided into server and client. The Alarm Server component on the server side monitors the status changes of the PV in the IOC, and sends data to Kafka based on the previous configuration and status of the PV. The server subscribes to the topic in Kafka, and displays the alarm interface on the Phoebus architecture: Alarm Tree, Alarm Table, Alarm Area Panel, Alarm Logger, and Annunciator. The functions are shown in Table 1.

Table 1: Alarm

Application	Function
Alarm Tree	Display hierarchical alarm data flow, record the type and severity of alarms
Alarm Table	Display real-time alarm information of process variables in tabular form
Alarm Area Panel	Display the name of the top alarm layer, which can be used to display the basic alarm status indication interface
Alarm Logger	Record all statuses associated with the service archive and the alarm server
Annunciator	Display the alarm time, alarm security level and the description of the alarm PV and send out the alarm voice

HARDWARE

Figure 3 is a diagram of the hardware architecture of the control system. The operator interface uses Linux operating system to run Phoebus to realize alarm visualization and complete the human-computer interaction process. IOC provides a physical interface for the device and communicates directly with the underlying device[2].

IOC uses records to limit the description of the controlled device, so that the value obtained by the controlled device is divided into alarm levels.LAN realizes the communication between OPI and IOC. IOC and OPI adopt CA (Channel Access) channel access mechanism。CA includes client CAC (CA Client) and server CAS (CA Server), which provide application interface libraries for OPI (Operator Interface) and IOC respectively[3]

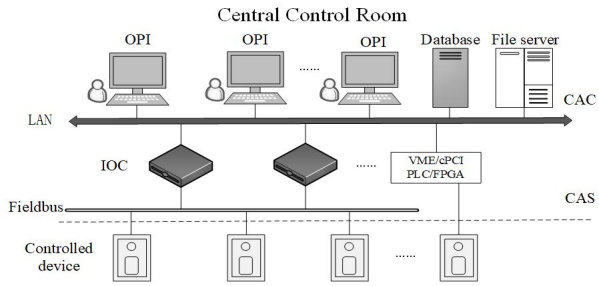


Figure 3: Hardware architecture of control system.

REAL-TIME REMOTE INTERFACE

Remote Real-time Interface Deployment

In order to realize the function of remotely accessing the interface of the central control room, the enterprise WeChat is adopted. At the same time, in order to ensure data security, the server and enterprise WeChat account are deployed to the intranet and the extranet respectively. The deployment process is shown in Figure 4. Intranet: The opi interface drawn by CS-Studio is deployed in the Tomcat application server on the intranet[4]. Write the configuration file config.php to connect to the enterprise WeChat, which mainly includes CorpId, TxlSecret and AppsConfig. Extranet: Deploy enterprise WeChat account, and log in to the enterprise WeChat account through the administrator's authorization for staff in the office. Enterprise WeChat obtains real-time data from the central control room through the firewall. The staff of the institute can view the real-time remote interface anytime and anywhere to understand the experimental situation.

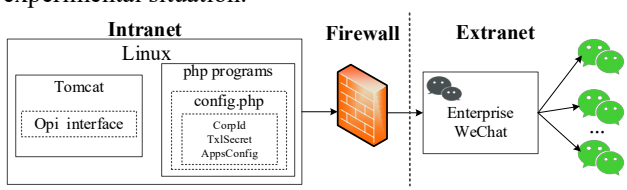


Figure 4: Remote real-time interface deployment.

Real-time Mobile Terminal Alarm Interface

Figure 5 shows the remote real-time data interface. The value of the alarm can be marked in red by setting the control rules of the OPI interface. The user group can view the relevant on-site real-time interface through the enterprise WeChat, which is helpful for the staff to understand the on-site operation in time and deal with the failure in time.

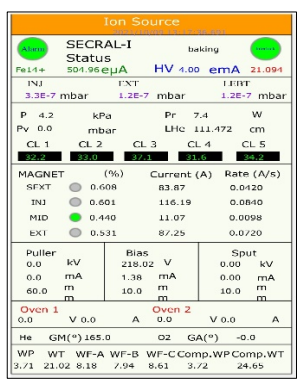


Figure 5: Real-time remote interface.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV006

CENTRAL CONTROL ROOM ALARM INTERFACE DISPLAY

Alarm

The alarm interface of Phoebus is shown in Figure 6. The alarm interface mainly includes four parts: Alarm Tree , Alarm Area Panel and Alarm Table[5].

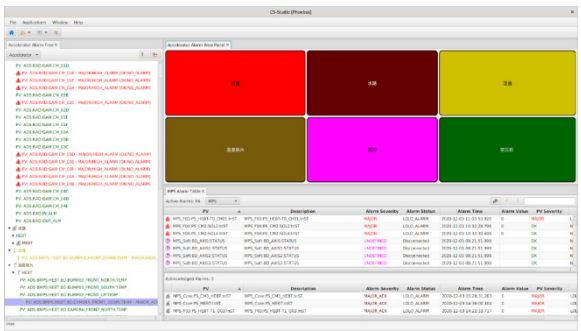


Figure 6: The alarm interface of Phoebus.

Alarm Tree

The alarm tree is mainly used to configure the alarm system, add PV and define the detailed information of the alarm. The tree-like hierarchical structure is used to record the PV that needs to be monitored. The tree structure mainly includes three parts: root, branch and leaf. Root represents the top-level domain name, branch represents the subsystem under the top-level domain name, and leaf represents the monitored process variable[6]. As shown in Figure 7, there are 6 top-level areas, namely dosage, waterway, flow intensity, temperature patch, vacuum and air compressor. Subsystems can be divided under each area, such as HEBT and MEBT in the waterway, and PV can be classified under the sub-system. The top-level domain name shows the most serious alarm status among all PVs under the domain name. The hierarchical view allows the operator to quickly find the affected subsystem or area.

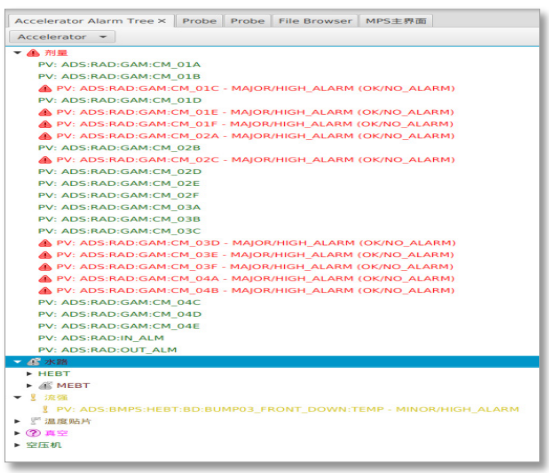


Figure 7: The interface of the Alarm Tree.

Alarm Area Panel

The Alarm Area Panel displays the most serious alarm level, and the panel color is colored according to the highest alarm in the area. Different colors represent different levels of alarms, and each alarm domain corresponds to the top-level alarm of the Alarm Tree. As shown in Figure 8, the red is the MAJOR alarm state, the red is the MAJOR alarm state and the operator confirms that the alarm is received, the yellow is the MINOR alarm state, the yellow is the MINOR alarm state and the operator confirms that the alarm is received, the purple is the INVALID disconnection state, and the green is the normal state.

Figure 8: The interface of the Alarm Area Panel.

Alarm Table

As shown in Figure 9, the Alarm Table is empty under normal conditions. When an alarm occurs, the alarm PV will be listed in the table, and the user can sort by PV name, description, alarm time and alarm level.

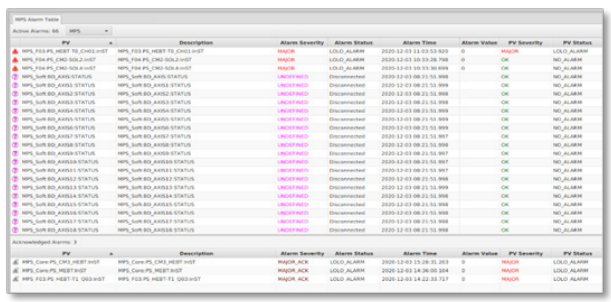


Figure 9: The interface of the Alarm Table.

Alarm Log Table

The Alarm Log Table records historical alarms and is updated every 10 s, which can be used to query historical alarm information, as shown in Figure 10.

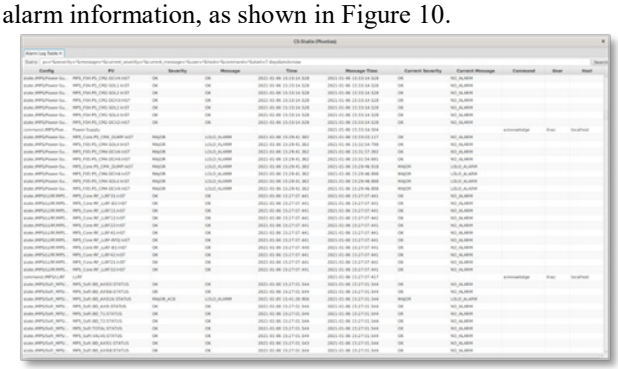


Figure 10: The interface of the Alarm Log Table.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV006

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Annunciator

When the PV alarm information that exceeds the threshold in the topic "AcceleratorState" is sent to the Alarm Server, the topic "AcceleratorTalk" will send the alarm to the Annunciator.The Annunciator interface will display the alarm time, alarm security level and the description of the alarm PV, as shown in Figure 11. By default, the central control room receives the alarm voice once/15 s, and the voice is the content of the description, and it broadcasts until the alarm PV returns to the normal threshold.

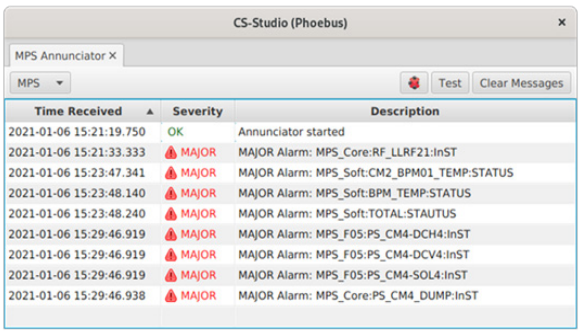


Figure 11: The interface of the Annunciator.

COMPARISON OF THROUGHPUT PERFORMANCE BETWEEN Kafka AND ActiveMQ

Kafka and ActiveMQ are streaming media platforms, both of which can process streaming data. The old version of the alarm system uses ActiveMQ. The alarm system of this report uses Kafka as an intermediate message plug-in. As shown in Figure 12 and Figure 13, JMeter is used to test the throughput of Kafka and ActiveMQ, both of which use a single thread to obtain 1000 data for multiple tests. The experiment shows that Kafka throughput is 6188119/minute, ActiveMQ throughput is 739153/minute. Kafka's performance far exceeds ActiveMQ. Kafka has high throughput and is more efficient for processing real-time data streams from accelerators. Moreover, Kafka can be used not only as a message intermediate plug-in, but also as a database cache data. There is no need to install the database separately, which simplifies the installation process. Therefore, it is more effective to use Kafka as the streaming platform of the alarm system.

Figure 12: The throughput of Kafka.

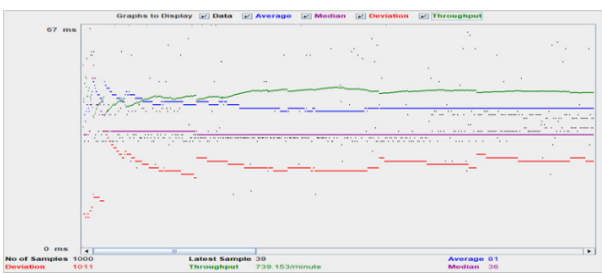


Figure 13: The throughput of ActiveMQ.

CONCLUSION

CAFe alarm system was built based on Kafka streaming media platform. Phoebus adds the PV to be monitored and displays the running status. Phoebus provides an interface for stable monitoring of PV status and real-time alarm function. Using Kafka can effectively increase the data throughput rate and ensure the high availability of the server. And use the remote real-time interface to improve the cross-regional alarm function.The entire control system has been deployed and put into operation in the central control room. The interface is simple and easy to operate, works well and is easy to maintain, which saves troubleshooting time and increases the maintainability of CAFe .

REFERENCES

[1] Xu Lunming, Wen Pengquan, Wang Jigang, Xuan Ke, Li Chuan, and Liu Gongfa, "The alarm system of Hefei light source based on BEAST", *Nuclear Electronics and Detection Technology*, vol. 35, 2015, pp. 417-421.

[2] C. Jianjun, Y. Youjin, Z. Wei, *et al.*, "Upgrade of Control System for 320 kV Heavy Ion Multidisciplinary Research Facility", *Atomic Energy Science and Technology*, cvl. 53, 2019, pp. 1612-1616. doi:10.7538/yzk.2019.youxian.0159

[3] Hao Peng-Hai, Xu Cheng-Long, and Liu Yi-Tian, "Monitoring and Alarm System for Power Grid Cloud Platform Based on Kafka and Kubernetes", *Computer Systems & Applications*, vol. 29(8), pp. 121-126, 2020. doi:10.15888/j.cnki.csa.007611

[4] Y. L. Zhang, K. J. Xue, F. Q. Guo, *et al.*, "Public Account Based Alarm Information Pub/Sub Pattern System for CSNS Accelerator Facility", *Nuclear Electronics & Detection Technology*, 2019.

[5] K.-U. Kasemir, X. H. Chen, and E. Danilova, "The Best Ever Alarm System Toolkit", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper TUA001, pp. 46-48.

[6] K. S. White and K.-U. Kasemir, "Alarms Philosophy", presented at 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09), Kobe, Japan, Oct. 2009, paper TUP022, unpublished.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV007

FAST CREATION OF CONTROL AND MONITOR GRAPHICAL USER INTERFACE FOR PEPC OF LASER FUSION FACILITY BASED ON ICSFF

Li Li†, Jun Luo, Zhigao Ni
Institute of Computer Application (CAEP) China Academy of Engineering Physics, MianYang, China

Abstract

Plasma electrode Pockels cell (PEPC) is the key unit of the multi-pass amplify system in laser fusion facility, whether the PEPC is effective determined the success rate of the facility experiment directly. The operator needs to conduct remote control and monitor during the facility is running, also can automatically judge whether the pulse discharge waveform is regular online. In traditional design and realization of control and monitor software, the fixed GUI cannot adapt frequent changes of the experiment requirements, and it will consume time and resources once more. We have designed a software framework (ICSFF) that loads all GUI widget elements related to control and monitor into board through plug-ins, and then by setting the respective properties, data source and built-in script of each widget achieve patterns like point control, flow control and other complex combined control, can also achieve data acquisition and varied display effects. It allows the operator drag and drop widget freely and configure the widget properties through the interface in a non-programming mode to quickly build the GUI they need. It not only apply to PEPC in facility, but also to other system in the same facility. ICSFF supports Tango control system right now, and more control systems will be supported in the future.

OVERVIEW

The control system of large-scale scientific experiment equipment is usually a heterogeneous system, including multiple subsystems with special functions, and PEPC of laser fusion facility is also a complex system, which contains a variety of hardware devices with different communication protocol interfaces. The typical large-scale control system like Figure 1. In the past, when developing remote integrated control system, we needed to design different interfaces for each different hardware protocol and design different operating interfaces for users. However, the solidified interface and interface design have great limitations. They are neither adapted to the frequently changing user operation requirements nor to the frequently changing hardware interfaces of experimental systems. Designers often need to spend a lot of time on the repetitive work of modifying and debugging code.

In order to change this situation, we have designed a software framework (ICSFF) that allows users to simply drag and drop and configure control properties through the interface in non-programming mode, and then quickly build the GUI what they need. This is not only applicable to the remote integrated control of the PEPC in facility, but also applicable to other systems in the facility.

† lili_top@163.com

FRAMEWORK INTRODUCTION

As mentioned above, we have developed a general control system framework for the integrated control system design of this large-scale facility, and provide editable functional modules for monitoring, control, data acquisition and storage functions with general requirements. The ICSFF software framework is a component-based distributed control system, mainly for non-real-time systems.

The software framework is suitable for the following network systems, and hardware devices are connected to the system through the network. The software of the control layer is deployed on the server or embedded controller to provide the control and data interaction of the hardware device. The monitoring layer software is deployed on the console computer to provide integrated operation, monitoring and other functions.

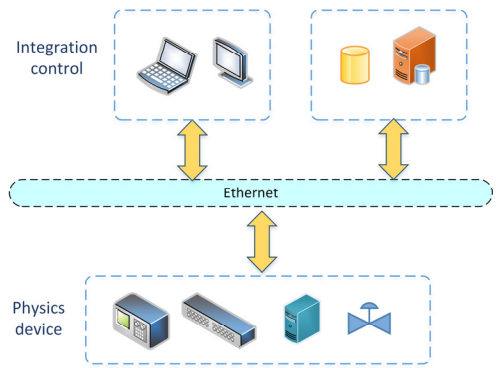


Figure 1: Typical large-scale scientific device control system structure.

The ICSFF software framework is divided into three layers: device service layer, system service layer and integrated monitoring layer [1]. See Figure 2.

The integrated monitoring layer provides a unified integrated operating environment for the control operation of the entire facility, enabling centralized control, monitoring and data management of the entire facility; the integrated monitoring layer can be divided into three different dimensions of integrated control functions according to actual needs: facilities, systems, groups.

The system service layer provides a combined control function for a single system under a bundle group, which is a large-scale control function aggregation of the device service layer, and this layer provides parameter delivery and experimental data archiving.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The device service layer provides software mapping for independent devices to implement device control, status acquisition, device diagnostics, and self-test, using device drivers. Therefore, all heterogeneous devices implement the unification of software interfaces and provide consistent standard services on network protocols [2].

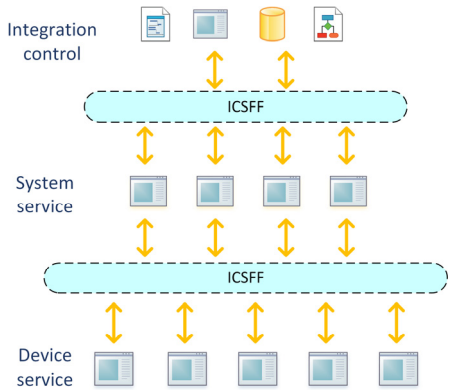


Figure 2: ICSFF software framework.

COMPONENT-BASED CONTROL AND MONITORING GUI DESIGN

Here we mainly discuss the GUI design of the software framework. To make the software framework have a flexible operation mode that can be configured at runtime, it is need to solve the problems of abstraction of the UI layer, separation of data and UI, and editability at runtime.

Direct interaction with users requires a variety of functional modules. Currently, our software framework provides login module, log module, panel module, service management module, scheme module, element module, and attribute editing module. A complete user integrated monitoring system can be built from these modules.

Element Module and Attribute Editing Module

Let's start with the smallest element module. At present, our element library contains 23 optional controls, including flow nodes, buttons, text edit boxes, multiple selection boxes, text displays, LCD digital displays, status lights, progress bars, Ring marquee, pump, valve, pipe, image, curve, time and other types. Currently supports Qt controls, Iocomp controls and Quc controls.

Each type of element contains its own different attributes, which include styles, data access interfaces, and data display styles. element properties can be edited. Users can edit the basic styles of the element such as the size, color, border, and font according to the needs of the display. Element style editing is shown in Figure 3, which shows the style editing interface of several element.

The purpose of separating the data I/O interfaces separately is to separate the data from the element, so that the software framework can adapt to more requirement. To configure the data in and out interface for each element, whether it is a button to send a command, or a status light, pump, etc. displayed by receiving data, you only need to select the interface provided in the corresponding service from the left side of Figure 4, and pass on the right side. Edit the script language file to realize the connection between the control and the data. The script language can also provide more complex calculations, so that the received raw data can be processed through calculations and then displayed on the element.

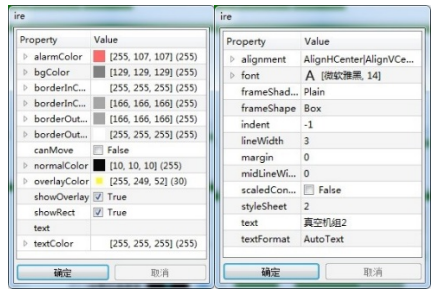


Figure 3: Element property editing interface.

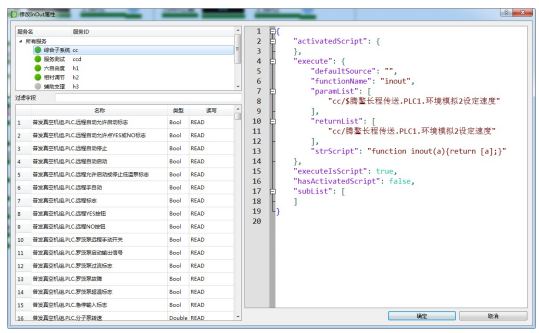


Figure 4: Data interface property configuration.

Data display style editing is shown in Figure 5, it realizes that certain controls, such as status light data, will undergo different color changes according to the received data. This is done by configuring keywords and values. Especially when the data types and data meanings defined by different manufacturers are different, users can uniformly set according to their own needs, which is conducive to the unification of the interface without changing the code.

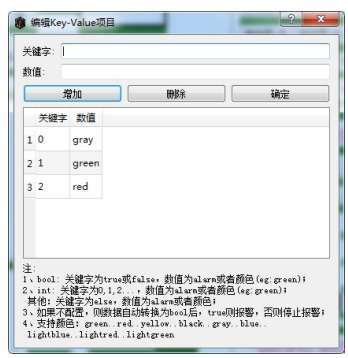


Figure 5: Data display style attribute configuration.

The data flow of the element is shown in Figure 6. The data input and the data received in the interface are all translated through the script language, and then sent and received by the FRTDB interface, which realizes the separation of the control and the data.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV007

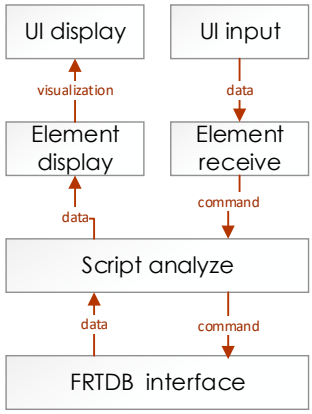


Figure 6: Data flow through element.

Panel Module and Scheme Module

A scheme is a container of controls, and a scheme can become a system interface with independent functions. The panel is a container for loading solutions. Programs provide users with the ability to build any system or subsystem, and programs can be nested. So, users can create multiple schemes to facilitate the use and management of the control system.

The panel provides program editing and running functions, it shown in Figure 7. User drags the elements into the schemes, and arranges the elements in the editing function provided by the panel. In order to realize the editability of the program, all the contents of the program need to be stored persistently, including the general attributes such as the position, size, and color of the control, as well as the configuration of the data in and out interface. The panel provides functions such as saving scheme to database, saving to file, loading scheme from file, and loading scheme from database. At the same time, it also provides convenient functions such as copying of controls, replacing control data in and out interfaces, so that users can create schemes more quickly.

Figure 7: Panel editing function.

Through the application model of this software framework, our development model has changed, the amount of code is reduced, the focus is more focused on the difficult points, the debugging efficiency is greatly improved, and the operation and maintenance cost are greatly reduced. Figure 8 shows our current working mode: after confirming the requirements with the users, confirming the interface requirements with various manufacturers, and then using the reuse of the ICSFF software framework to build the control system, and directly convert the fully reusable parts such as login, log, panel, etc. Utilize, some reusable parts such as program editing, attribute editing, etc., are changed according to user needs, and some functions and elements that are not available in the framework need to change the framework code or add new elements to meet user needs, and at the same time Reverse enriches the connotation of the software framework. After the iteration of unit testing and integration testing, the control system has completed the basic creation.

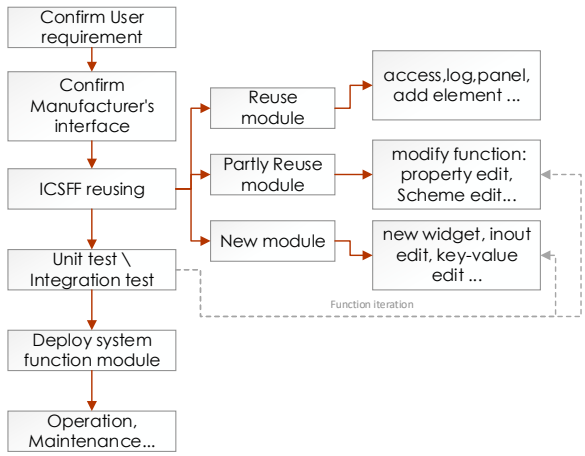


Figure 8: Software framework reuse process.

APPLICATION

The application of ICSFF in the software framework in the plasma switch control system is shown in Figure 9. As described in the previous chapter, the system framework is built through the software framework, and the required controls are added, and the properties and control data in and out of the interface content are edited. Complete the system. Compared with the actual time of our implementation and deployment, the previous system debugging and code modification took 1~2 man/month, while the current model only needs 0.5 man/month to debug and modify.

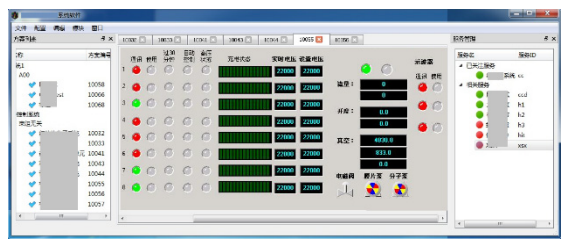


Figure 9: PEPC control system.

CONCLUSION

The ICSFF software framework contains many other components. This article only shows the application in the user interface. The goal of the software framework is to provide an open, scalable and reliable integrated control platform that can provide long-term operation and maintenance capabilities for various large and small systems and scientific experimental devices. In the future, we will conduct more exploration and research on the intelligent direction of the software framework, and add more intelligent modules to promote the current growing demand for data analysis.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV007

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] Z. G. Ni *et al*., "The Design of Tango Based Centralized Management Platform for Software Devices", in Proc. *ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1121-1124. doi:10.18429/JACoW-ICALEPCS2017-THBPL0

[2] Z. G. Ni *et al*., "The Design of Intelligent Integrated Control Software Framework of Facilities for Scientific Experiments", in Proc. *ICALEPCS 2019*, New York, NY, USA, Oct. 2019, pp. 132-136. doi:10.18429/JACoW-ICALEPCS2019-MOMPL007

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

WEB GUI DEVELOPMENT AND INTEGRATION IN LIBERA INSTRUMENTATION

D. Bisiach†, M. Cargnelutti, P. Leban, P. Paglovec, L. Rahne, M. Škabar, A.Vigali
Instrumentation Technologies doo, Solkan, Slovenia

Abstract

During the past 5 years, Instrumentation Technologies expanded and added to the embedded OS running on Libera instruments (beam position instrumentation, LLRF) a lot of data access interfaces to allow faster access to the signals retrieved by the instrument. Some of the access interfaces are strictly related to the user environment Machine control system (EPICS/TANGO), and others are related to the user software preferences (Matlab/Python). In the last years, the requirement for easier data streaming was raised to allow easier data access using PC and mobile phones through a Web browser. This paper aims to present the development of the web backend server and the realization of a web frontend capable to process the data retrieved by the instrument. A use-case will be presented, the realization of the Libera Current Meter Web GUI as a first development example of a Web GUI interface for a Libera instrument and the starting point for the Web GUI pipeline integration on other instruments. The HTTP access interface will become in the next years a standard in data access for Libera instrumentation for quick testing/diagnostics and will allow the final user to customize it autonomously.

INTRODUCTION

In the accelerator environment, the data access is usually performed with well-established and standard access using EPICS, TANGO, Labview, and Matlab interfaces. This software does not allow only access to data but also integrates the control system and safety interlocks to prevent damage to the accelerator block unit.

The drawback of this reliable and safe type of implementation is that any new instrument that is placed in the accelerator environment needs to be configured on the server-side and usually this procedure requires time and effort since the connection is not immediate.

The need for a faster way for evaluation and testing of new devices triggered the attention to the Web interfaces that were already developed in-house by the Red Pitaya project [1] that integrates a fast and easy to access interface that can be used for quick data acquisition.

INTERFACE BETWEEN INSTRUMENT AND APPLICATION SOFTWARE

The access to the setting and the data provided by the instrument is allowed by the software structure reported in Fig.1. The lower layer is tightly bonded to the hardware interface and is responsible to communicate at a lower level with the FPGA and the CPU processes. The second layer called Machine Control Interface (MCI) connects all the user interfaces by providing APIs that enable the servers to access the configuration parameters, the status information, and the data acquired by the instrument. The server-side applications expose through the network to the client-side different application protocols: libera-ioc (EPICS), libera-ds (TANGO), libera-telnet (Labiew/Matlab), libera-cli (user access with the bash) and, as a new feature described in this paper, the libera-http interface that enables the Web access.

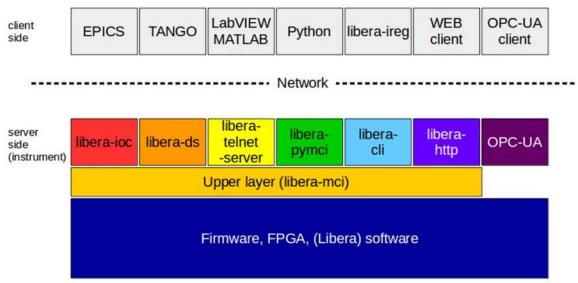


Figure 1: Application stack layer.

Some of these protocols can run in parallel (e.g. EPICS, bash, and HTTP) which makes the troubleshooting of the instrument much easier and more efficient.

HTTP APPLICATION ARCHITECTURE BASED ON REST API

As mentioned in the previous paragraph, access to the instrument can be performed by any device that can access the same network using the HTTP protocol. A typical use case is reported in Fig. 2 where the instrument is accessible using the wired and wireless network:

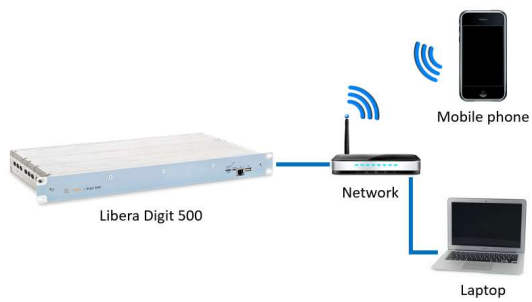


Figure 2: Access to the instrument in a local network.

The system architecture is based on the REpresentational State Transfer (REST) software architectural style and provides the services through Application Programming Interfaces (API) that allow the programmer to easily implement access to the instrument internal interfaces.

The server interface starts during the boot of the instrument and opens a port that is accessible to the other devices

† danilo.bisiach@i-tech.si

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

connected to the same network environment. The Libera instrument acts as a server, and any local device acts as a client. The communication API uses the JavaScript Object Notation (JSON) to exchange the data between the server and the client is reported in Fig. 3 and Fig. 4:

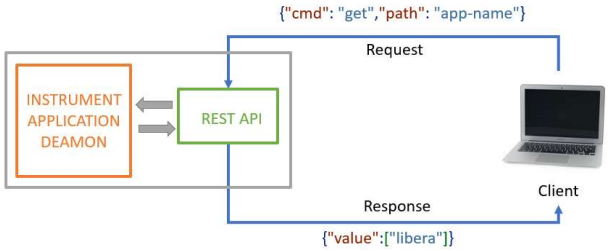


Figure 3: HTTP REST API application architecture.

The requests are performed on the client-side by sending desired parameters in JSON format to the REST API. The server will then return the requested node in JSON format that will be processed by the client and presented in a human-readable format on the WebGUI.

Figure 4: HTTP REST API application architecture node detail.

The integration of the HTTP server on different instruments is almost immediate since the building of the application is fully automated by the existing revisioning and building system. Different instruments, so different software applications, can share the same code for the server application and this feature allows high scalability of the server application.

The only part that needs to be customized development for every new application is the WebGUI code which includes the path to the application nodes. All the settable nodes on the instrument are specified by the node-set labels where the path of a specified node can be accessed as reported in Fig. 5:

The node-set tag is composed by the name we want to give to the variable in the WebGUI, by the path label that is representing the target node in MCI application and allows access to a specific settable variable or a data stream on the instrument, and by the polling-period which specified the refresh time in the WebGUI application.

```
<div class="ireg-node-container">
  <div class="ireg-node-set-name">
ATTENUATION
    </div>
  <tablev>
    <node-set name="Channel 1" path="application.hk.attenuation.Ch1" :pooling-period=1000></node-set>
    <node-set name="Channel 2" path="application.hk.attenuation.Ch2" :pooling-period=1000></node-set>
    <node-set name="Channel 3" path="application.hk.attenuation.Ch3" :pooling-period=1000></node-set>
    <node-set name="Channel 4" path="application.hk.attenuation.Ch4" :pooling-period=1000></node-set>
  </tablev>
</div>
```

Figure 5: Code snippet with the implementation of a node readout from the system.

HTTP APPLICATION ARCHITECTURE BASED ON WEBSOCKET

An additional high-performance interface based on WebSocket technology was integrated into the Libera software.

The interface is compliant with the RFC 6455 and compatible with the HTTP protocol by allowing a full-duplex communication between the client and the server running without the need to re-establish the communication at every request but by keeping it open to facilitate the real-time data transfer and data streaming over TCP. Figure 6 reports the architecture implemented with the WebSocket architecture:

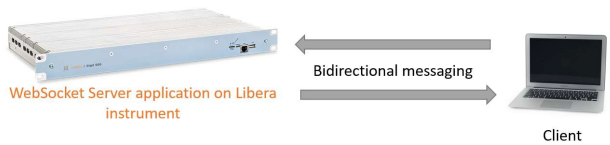


Figure 6: WebSocket application architecture.

The main benefit of this implementation is to allow a maximum performance on data exchange with minimum overhead, and this can make the difference with the REST/API.

It was chosen for the implementation of the 12 Channel Libera Current Meter WebGUI interface which can handle and plot the data acquired from the instrument with minimum latency and high robustness.

EXTENSION AND CUSTOMISATION OF THE HTTP USER INTERFACE: DEVELOPMENT AND INTEGRATION OF A 12 CHANNEL CURRENT METER

One of the main benefits of the HTTP implementation is to allow the final user to customize and integrate the application.

The extensions of the WebGUI can consist of the simple addition of buttons/graphical effects or also of some more consistent improvements. The realization of the 12 Channel Current Meter GUI interface was a very extensive customization of the user software that consisted in the implementation of really demanding features listed below:

1. Capability to perform calculations on the data acquired such as mean values and standard deviation for the acquired signals.
2. Continuous backup of the acquired data in .csv file format.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

3. Integrate the data acquisition from 3 independent 4-channel acquisition instruments as a 12 channel acquisition instrument.
4. The ability of the WebGUI to run smoothly for 1-month operation, without any loss of data and by keeping the measurement going during the operation.

These features required a high number of additions, testing, and revision of the actual HTTP server implementation. Due to these features, the decision was to implement all the communication based on WebSockets.

All the 3 integrated instruments were acting as an independent device with 12 channels by maintaining a constant synchronization between the 12 channels. The implemented architecture is reported in Fig. 7 where one 4-channel Libera Current meter unit is acting as the master unit which includes the HTTP server on which the client will access the acquired data. The two other 4-channel Libera Current meter are synchronized with the master unit and continuously provide the acquired data stream:

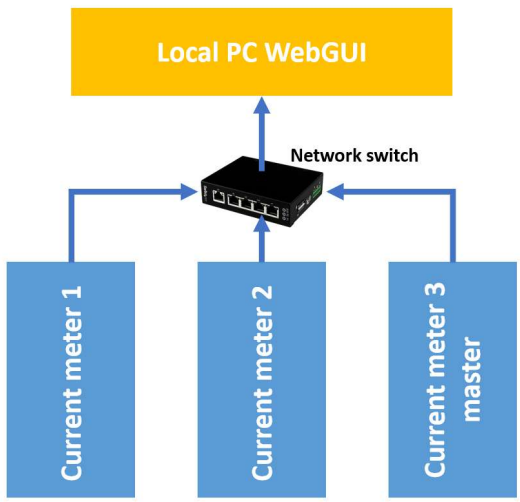


Figure 7: 12 Channel Libera Current meter integration. 3 independent instruments are acting as one on the WebGUI interface side.

The WebGUI screenshot of Fig. 8 reports the final result of the implementation.

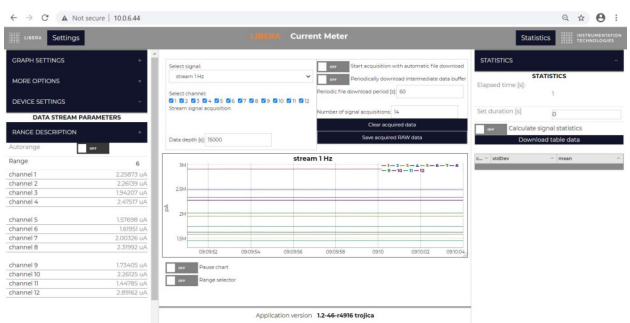


Figure 8: 12-Channel Libera Current meter integration WebGUI.

The most critical point was the testing of the user interface by checking the ability to run smoothly for one month time. These tests were performed by adding to the application software many additional data exchange requests and to stress the memory consumption of the Web browser. The interface was demonstrated to work smoothly for 30 consecutive days with this additional workload on Microsoft Edge (based on the Chromium engine). The main parameters were constantly monitored using the Web browser diagnostics features such as the task manager and the performance options in the Web browser developer mode. The use of the WebSocket architecture allowed the interface to run smoothly even if a lot of data were continuously accumulated and plotted.

As a first experience and proof of concept about the use of WebGUI interfaces based on Javascript for instrument data acquisition, the decision is to integrate the Selenium plugin for Python into the future projects to automate the GUI regression tests and scalability during the QC operations.

RESULTS AND CONCLUSIONS

The development of an HTTP REST API and WebSocket interfaces required a lot of effort in terms of system implementation, integration, testing, and debugging. One of the main benefits is that such a system can be easily accessed by any final user without the need for any control system software (EPICS, TANGO) or proprietary software (Matlab, Labview) already running on the data acquisition system. These features made it very reliable and the first choice of use during system troubleshooting and installation.

The other benefit was to introduce high scalability of the interface: many instruments can benefit from this implementation since the HTTP server is available as a quick add-on for all the Libera instruments with minimal effort dedicated to porting. The performance of the WebSockets can make a difference when a large amount of data needs to be retrieved from the instrument.

Another future implementation within the HTTP or WebSocket can be the GRPC [2] Node-based implementation, which will unify high performance in an easy-to-use platform.

As an ending note, we can confirm that the developed HTTP REST API and WebSocket interface has now became one of the first choice data access interfaces for the testing of the instruments.

ONLINE SOURCE

[1] RedPitaya website, https://redpitaya.com/
[2] GRPC framework, https://grpc.io/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV010

SCALING UP THE ALBA CABLING DATABASE AND PLANS TO TURN INTO AN ASSET MANAGEMENT SYSTEM

I. Costa†, A. Camps Gimenez, R. Cazorla, T. Fernández Maltas, D. Salvat*
ALBA-CELLS Synchrotron, Barcelona, Spain

Abstract

The "Cabling and Controls Database" (CCDB) is a central repository where the different teams of ALBA manage the information of installed racks, equipment, cables and connectors, and their connections and technical specifications. ALBA has modernized this web application for sustainability reasons and fit new needs detected throughout the last years of operation in our facility. The application has been linked to Jira to allow tracking problems in specific installed equipment or locations. In addition, it also connects to the ALBA Inventory Pools application, the warehouse management system, where the stock of physical equipment and components are maintained to get information on the life cycle of the different devices. These new features, integrated with proprietary products like Jira and Insight, aim to become ALBA's asset management system. This paper aims to describe the main features of the recent application upgrade, currently in continuous development.

INTRODUCTION

During the ALBA's design phase back in 2006, the Management Information Systems section (MIS), under the Computing Division, started to develop the "Cabling and Controls Database" (CCDB) [1]. Since then, this web application has been used as a central repository to keep information of all racks, equipment, connectors and cables used in ALBA. At its early stage, the application was very useful for the cabling tender and the installation phase. Subsequently, other features were implemented in order to ease the maintenance of the Tango Control System [2] and the Equipment Protection System (EPS) [3] by the Controls group, who gets the data through an API or directly from the web interface.

This in-house development has been evolving for years, and in 2019 started a process of technological upgrade. In this upgrade new features have been included, such as the integration between equipment instances and Jira, and also their integration to the ALBA Inventory Pools system. These integrations aim to become a new ALBA's Asset Management System.

CCDB UPGRADE

In origin, the "Cabling and Controls Database" was integrated in the ALBA's Intranet and built in python Plone/Zope which acted as the web interface to ease the visualization of the data stored in a MySQL database. The CCDB provides also an API, which has also been upgraded to a RESTful system.

Along the last years of usage, the Electronics section of ALBA, managers of the application, detected new needs and proposed new features to be developed. Without a first modernization of the technology these new features could not be developed.

Technological Upgrade

Starting on January 2019, the original CCDB application was redefined and migrated to Django Framework [4], a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern. Like in the old application, the new database is also a MySQL.

This upgrade was necessary to decouple the application from the outdated Plone ALBA's Intranet, following the tendency of the other applications developed by the MIS section. In the same direction, the new application's look and feel meets the style guide of the MIS' applications, based on Bootsrap [5] (see Fig. 1).

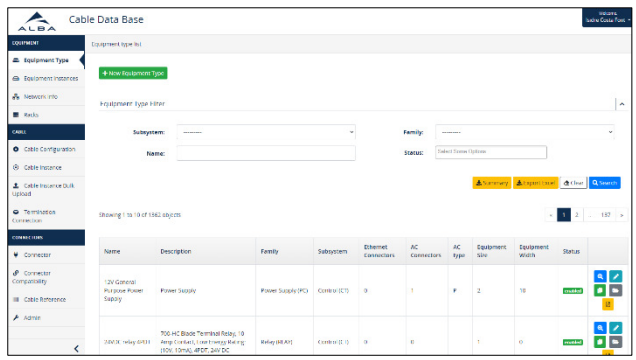


Figure 1: View of the CCDB web application.

To carry out the migration, the new application was developed from scratch and, after a period of coexistence between both the old and the new system, finally on February 2020 all the data was migrated to the new CCDB Django application and the old version was disabled.

New Features

Among the new features introduced after the migration of the application we should highlight two of them, that enable the application to be part of a future Asset Management System.

The first of these features requested by Electronics was the tracking of equipment instances and location using Jira, our issue tracking system. This feature allows to detect dysfunctional equipment or hot locations.

† icosta@cells.es
* on leave

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV010

The other requirement was the interconnection between the CCDB and the ALBA's Inventory Pools application, our warehouse management system. This integration gives users the opportunity to inform which physical equipment from the warehouse is installed in the equipment instance of the CCDB.

ISSUE TRACKING OF EQUIPMENT

The Concept of Equipment Instance

An equipment instance is an object created in the CCDB that represents an instance of an equipment type model. The equipment type is a template that contains a set of channels, each channel has associated a connector type that has already been created in the CCDB [1]. Furthermore, every equipment type has associated its documentation, like technical specifications and manuals.

When a user creates a new equipment instance in the CCDB, he/she is virtually placing an equipment of the chosen type in a particular location of the facility. In addition to the location and the type, user should select a system and subsystem from a list, to add a functional attribute to the equipment instance. The result of this combination of fields is an equipment code, which will be a unique identifier in the whole CCDB. In Fig.2 there is an example of equipment instance code and how it is structured.

SR-CT-CPCI-RKA04B01-01

System-Subsystem-Family (Eq.Type)-Location-Code

Figure 2: Example of equipment instance code.

Jira Service Desk and Insight Application

In 2016 ALBA adopted Jira as its Service Management System, given that it fulfils all the needs of the organization, not only in terms of service but also in terms of project management [6]. Since the implementation of this commercial software the creation and management of different types of issues to different Service Desk teams has been centralized in a unique Customer Portal. With the support of the MIS section, every team has configured their workflows and processes according to their needs to handle their service desk requests.

As it is well known, there are a large number of plugins and applications in the market that can be installed in Jira. Some teams of ALBA started to use an application called Insight, which is fully integrated in the Jira Service Management. Insight is used for asset and configuration management [7] and it is very versatile, since any organization can completely configure its schema according to their needs.

Due to the high accessibility of the API provided by Jira, it is not difficult to make interact our own applications with Jira or Insight. However, in order to centralize all these interactions, the MIS section developed an internal REST API built in Python called jira-rest-service.

Integration of CCDB and Jira

Regarding to the equipment instances, one of the new requirements raised by the Electronics group was to allow users to create Jira issues from the CCDB web application relating directly the equipment instance. This feature enables the detection of dysfunctional equipment or hot locations.

The Jira issues are listed in every equipment instance detail page (see Fig. 3), or are easily reachable directly in the Jira Service Desk, where users can manage the issues at their convenience (see Fig. 4).

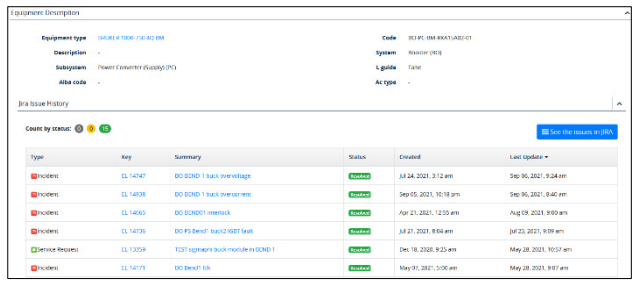


Figure 3: Equipment instance detail page with the list of its related Jira issues.

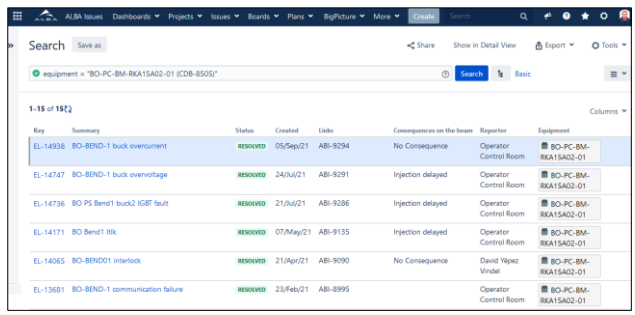


Figure 4: Jira Service Desk search page filtering by the Equipment instance showing the list of issues.

To allow this integration between CCDB and Jira, an object schema in the Insight application had to be created and configured. This schema is composed by all the possible location places that could occupy an equipment instance in CCDB, formed by combinations of fields location, number, row id and row position. Furthermore, every equipment instance created in CCDB, currently 8764 elements, are created also in Insight and related to its location place. When a new equipment instance is created in CCDB, the internal jira-rest-service API is requested to create its equivalent object to Insight, keeping the integrity of the data between both systems.

This equipment instance object of Insight is fully accessible from Jira Service Desk, since it has been defined as a custom field and added to the screens of the Electronics group Service Desk.

INVENTORY POOLS SYSTEM

The ALBA's Inventory Pools is an in-house web application developed by the MIS section that allows the different groups of ALBA the management of the inventory of their items as well as the booking and lending of the material by other users.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Technologically, it is built in a back-end / front-end architecture, using Django and AngularJS that interacts each other through the back-end REST services.

In a nutshell, Inventory Pools gives every group of the organization the opportunity to create their own pool, where they can manage their catalogue of materials, stocks and locations in the different warehouses or labs in the facility. To manage this information in the pool, every group assign different roles to particular users, such pool managers or pool operators, to be able to do different actions according to their responsibilities.

Booking System

Every pool can be public or not, depending if their items can be borrowed by any member of the ALBA staff. This is one of the main features of this application, indeed this was its original motivation. Figure 5 shows a page of the booking system.

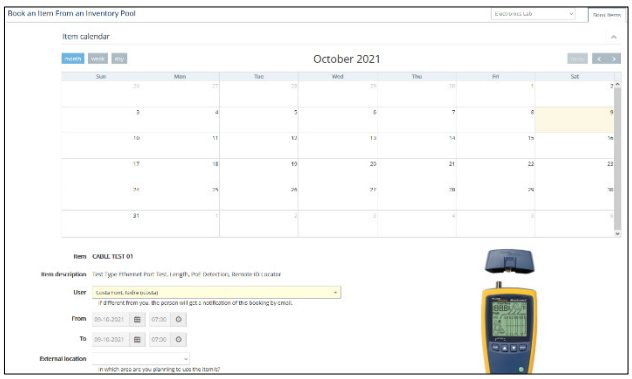


Figure 5: View of the Inventory Pools web application.

Warehouse and Inventory Management

To ease the management of the material stock, managers and operators of every pool can set up the locations in the system where the items are actually placed in their warehouse.

Every item of the pool has its unique barcode, provided by the system or by the pool manager. This barcode can be printed in a label and posted to the item or to its location. Teams are free to let users to get the material directly from their warehouse. In this case, a set of devices (touchscreen, barcode reader and an access card reader) controlled by a raspberry pi is installed in the warehouse (see Fig. 6). User should pick up the item, read the barcode and the item will be displayed in the system. After authenticating himself with his/her ALBA's access card through the card reader he/she can get the item recording it automatically to the system. An option to search the location of the item is also available as well as an option for the pool members to manage the material directly in site.

INSTALLATION OF ITEMS IN CCDB

As it is explained before, an equipment instance of CCDB is the representation of the specific location of the facility that will be used to place an equipment of a particular type for a determined purpose or functionality.

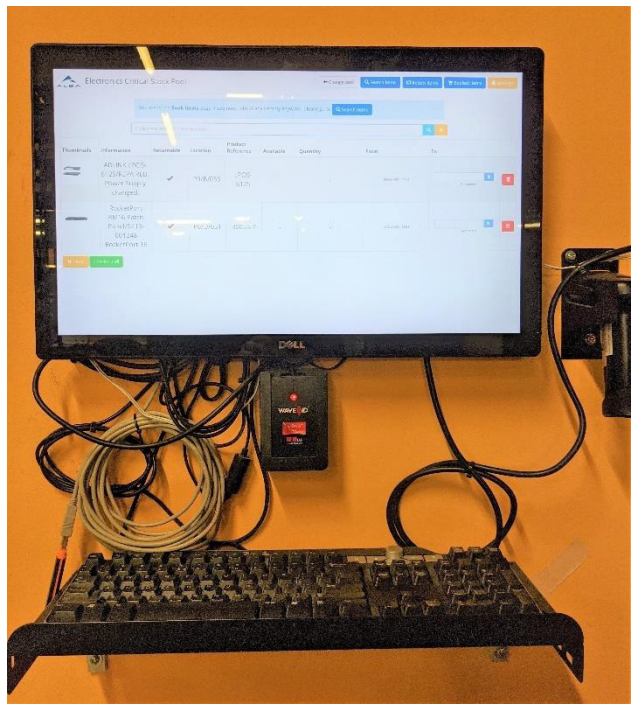


Figure 6: Set of devices installed in the ECS warehouse.

But until now, the specific physical device with its own barcode, that will occupy that equipment instance is not known. This is the need that the Electronics group detected throughout the last years of operation in our facility.

Integration of Inventory Pools and CCDB

To fulfil this need, the MIS section has customized the generic Inventory Pools application in order to allow the interconnection of the Electronics Critical Stock (ECS) with CCDB. Thus, users can install physical devices or components from Inventory Pools to CCDB.

If a pool, like ECS, is configured to interact with CCDB some differences are applied in its features. The barcode field becomes the "Alba Code" field, with the same purpose of identifying the item. A new field "Equipment Instance" is included for each item in the list and filter page.

To install the item in the equipment instance, user is able to do it in the equipment instance of CCDB or alternatively using the warehouse feature of the Inventory Pools. For that purpose, user can go to the warehouse of ECS take the item from its location, read the barcode and inform the system that he/she is going to install it physically in the selected equipment instance of CCDB (see Fig. 7).

When user performs this operation, the item becomes unavailable for other users, because it is being used in the installation. Besides, the managers and operators of the pool can see the code of the equipment instance where the item has been installed in the list of items.

From the CCDB point of view, users can do the same operation, getting a list of items that can be installed in the equipment instance. It is allowed to install multiple items in the same instance, because an equipment instance can be composed by multiple components. In fact, there is a new feature in CCDB, still to be developed, to give the

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV010

possibility of splitting the equipment instances in sub-equipment. Doing this, users would be able to install all the components one by one in each real location within an equipment instance.

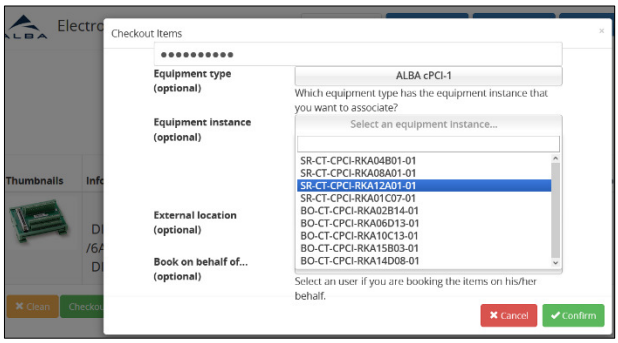


Figure 7: Example of how to select an equipment instance from the warehouse using the Inventory Pools application.

Asset Product Object in Insight

In the integration between CCDB and Inventory Pools the same CCDB object schema of Insight has been used. A new object called Asset Product has been created. When a manager or operator of the ECS pool creates a returnable item, this item can be created to Insight as an Asset Product. Internally, this time from Inventory Pools, the jira-rest-service API is requested to create the new object in Insight. The only data necessary to do it is the Alba Code and the item's name.

An item is considered returnable when it is a non-fungible material, when it is unique and identifiable with its own serial number. An Alba Code will be assigned to it in our system, which will be the code used to be identified also in CCDB.

When an item is installed in an equipment instance, regardless where this operation is being done, from CCDB or Inventory Pools, the jira-rest-service API will relate both objects in the Asset Product of Insight. On the opposite, an item can be uninstalled from the equipment instance in CCDB, in this case our API will remove their relationship. In Insight, an historic of events is stored for every object, so a tracking of operations in the Asset Product can be obtained.

ASSET MANAGEMENT

The integration between both applications, CCDB and Inventory Pools with Jira and Insight can be considered as a first step to build a new Asset Management System, since the combination of these systems covers all the four aspects that every asset needs to be unique (functional, product, geographical location and life cycle) (see Fig. 8).

Functional Aspect This aspect gives information about what is the functionality or purpose of the asset. Normally, an asset belongs to a system and sub-system that can be translated as its functional position. This is a mandatory information for the creation of the equipment instances in CCDB.

Product Information about what type of product the asset is. Types can be grouped by families to ease the categorization of products. Also, the serial number or barcode, that allow the identification belong to this aspect. Stored in both CCDB and Inventory Pools there are information and documentation about the item type. While the barcode, the serial number and other manufacturer information are maintained in Inventory Pools.

Geographical Location Regarding to this aspect, when an equipment instance is created in CCDB, user should provide a location of the facility, formed by the combination of fields location, number, row id and row position. This combination is the part of the equipment instance code related to its location (i.e. RKA12A01).

Life Cycle The integration with Jira can help users to manage the life cycle of items, keeping the history of changes or problems throughout its usage in the facility.

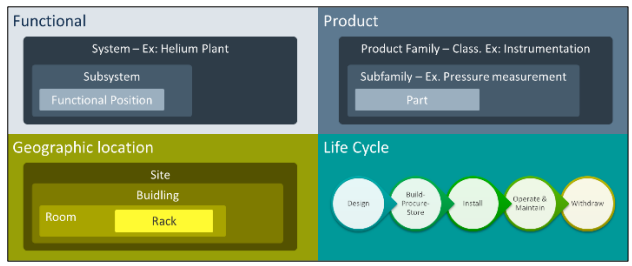


Figure 8: The four aspects of an asset [8].

Future Roadmap

The base of this incipient asset management system has been implemented successfully, but there are new features that need to be developed in order to complete it.

First, a definition of the process of a standard life cycle of an asset is needed to be implemented in Jira and Insight. Now, users can only install or uninstall items in the equipment instance, but the life cycle is much more than this.

And also, the multi-equipment feature in CCDB is needed to provide a complete break-down of components in each equipment instance.

CONCLUSION

The "Cabling and Controls Database" (CCDB) has been migrated to a new system for sustainability reasons, but also new features have been implemented. One of them is the integration with Jira in order to allow users the issue tracking of the equipment instances which can ease the detection of dysfunctional equipment or hot locations. Other remarkable issue is the integration with the Inventory Pools application, which enables the installation of physical equipment to the CCDB equipment instances.

These integrations are the first steps to build a new asset management system that combine the functional and geographical information provided by CCDB, the detailed information of the products from items of Inventory Pools and the tracking of the asset extracted from Jira and Insight.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV010

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

There are still some features needed in order to make it more functional. Since the migration was finished, CCDB is in continuous development, not only from the asset management point of view.

REFERENCES

[1] D. Beltran *et al.*, "ALBA Control & Cabling Database", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, October 2009, pp. 423-425, paper WEP009.

[2] S. Rubio-Manrique *et al.*, "A Bottom-up Approach to Automatically Configured Tango Control Systems", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, October 2011, pp. 239-241, paper MOPMN003.

[3] A. Rubio *et al.*, "ALBA Equipment Protection System, Current Status", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, October 2017, pp. 1599-1603. doi:10.18429/JACoW-ICALEPCS2017-THPHA096

[4] The Django Framework, https://www.djangoproject.com/

[5] Bootstrap, https://getbootstrap.com/

[6] M. Martin *et al.*, "Streamlining Support and Development Activities Across the Distinct Support Groups of the ALBA Synchrotron with the Implementation of a New Service Management System", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, October 2017, pp. 298-303. doi:10.18429/JACoW-ICALEPCS2017-TUMPL02

[7] Insight Application, https://confluence.atlassian.com/servicemanagementserver/getting-started-with-insight-1044784303.html

[8] M. Martin, "Learnings from CERN on Asset Management", Internal report, November 2018.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV011

NOTIFICATIONS WITH NATIVE MOBILE APPLICATIONS

B. Bertrand*, J. Forsberg, MAX IV, Lund, Sweden
E. Laface, G.Weiss, European Spallation Source ERIC, Lund, Sweden

Abstract

Notifications are an essential part of any control system. Many people want to be notified of specific events. There are several ways to send notifications: SMS, e-mails or messaging applications like Slack and Telegram are some common ones. Those solutions frequently require some central configuration to record who will receive messages, which is difficult to maintain. ESS developed a native mobile application, both for iOS and Android, to manage notifications. The application allows the users to subscribe to the topics they are interested in, removing the need for a central configuration. A web server is used as gateway to send all notifications following Apple and Google protocols. This server exposes a REST API that is used both by clients to send messages and mobile applications to retrieve and manage those messages. This paper will detail the technical implementation as well as the lessons learnt from this approach.

INTRODUCTION

The European Spallation Source (ESS) is under rapid development in Lund, Sweden. More and more parts of the control system are put into place, which means a growing number of messages and alarms are triggered. People want to be notified to keep track of what is happening and to know if any action is required. There were several ways to send notifications depending on the application. IT had a service to send SMS, some applications were relying on e-mails. We also developed Telegram and Slack bots. The Telegram bot was quite popular as users could check messages on their phone via the native mobile application. The issue was that the configuration was centralized: each new user had to be added manually with the list of topics he was interested in. That was demanding to maintain and didn't scale well. The other problem was the inclusion in existing applications which wasn't trivial. We wanted a general purpose solution that would be easy to use from any application and would unify the user experience.

CONCEPTS

We wanted users to be able to subscribe themselves to the notifications they are interested in. This would remove the need for a central configuration that is laborious to manage. To achieve this, notifications have to be grouped in categories named *services*. The number of *services* doesn't have any limit in theory. In practice, too many services would make it difficult to decide which to subscribe to. With too few services, there is a risk that the user will receive many unwanted notifications. Some services used at ESS are called Logbook On-Call, Logbook TS2, OpenXAL and Prometheus CSI. Those examples show that a service can be linked to an application (like OpenXAL), but that one application can also send messages to different services (like the LogBook).

Smartphones are part of an infrastructure that makes it easy to dispatch notifications. We chose to develop a specific mobile application that we could customize to our need. This application would be used to subscribe to the *services*, receive notifications and read messages. Two clients are available, one for Apple iOS devices [1] and one for Android users [2].

Sending notifications to a mobile phone can be done relying on Apple and Google infrastructure. We decided to design a REST API that is used both to communicate with the mobile clients and to forward the notifications received from the system. Sending a notification only requires a POST to this central server, named Notify server [3], making the integration in existing application simple.

NOTIFY SERVER

The Notify server was developed with FastAPI [4], an async Python web framework, that quickly became very popular in the past years. It is based on Starlette [5], a lightweight ASGI framework, and Pydantic [6], a data validation library using python type annotations. PostgreSQL [7] is used as database. FastAPI was designed to make writing API easy and is based on OpenAPI standard [8]. It automatically generates an interactive API documentation with exploration via Swagger UI [9]. This interface, showed in Fig. 1, is used by admin users to perform basic operation, like creating new services.

Message Sending

As described earlier, a notification has to be linked to a *service*. Only admin users can register a new service. When creating a new service, an UUID is generated and used to identify the service. This id has to be used by clients to send a notification associated to that service. This is done by sending a POST to */services/{service_id}/notifications* with the fields *title*, *subtitle* and *url* in the body. Only the title is required. Other fields are optional. The subtitle usually contains a longer description. The URL can be used to redirect to a specific website, like a LogBook entry, to have more details. This link will make the messages clickable in the mobile app. Figure 2 and Figure 3 show how to send a notification using curl or Python.

As there is no authentication, a filtering based on IP address is performed to avoid receiving messages from an untrusted source. Once received, the message is forwarded to all users who subscribed to this service using Apple and Google push infrastructure.

* benjamin.bertrand@maxiv.lu.se

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

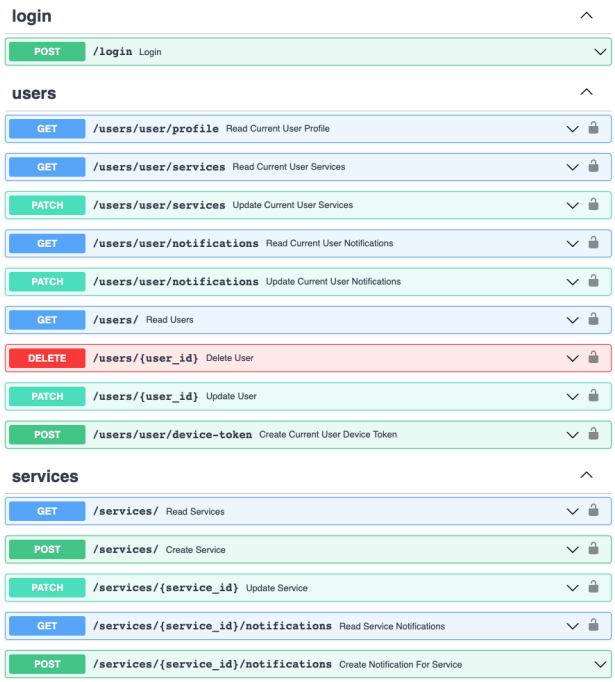


Figure 1: Notify server API

```
curl -X 'POST' \
  'https://notify.maxiv.lu.se/api/v2/services/f848b451-e3c8-497b-a6da-2cb571d9d95e/notifications' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "My message",
  "subtitle": "This is a test",
  "url": ""
}'
```

Figure 2: Sending a notification with curl

```
data = {"title": title, "subtitle": subtitle, "url": url}
response = httpx.post(
    f"{server}/api/v1/services/{service_id}/notifications", json=data
)
```

Figure 3: Sending a notification from Python

Communication with Mobile Clients

Mobile clients need to authenticate to access the REST API. To login, username and password are sent to the server and checked against an Active Directory service. If the credentials are correct, the server sends back a JSON Web Token (JWT). This token shall be included in the header of any future request. It is valid for a limited number of days and the user will only be forced to authenticate again when it has expired. This solution allows to not store any credentials in the phone or server.

Once logged in, the mobile app can use the API endpoints to get the list of services, subscribe or unsubscribe and read notifications linked to the current user. For notifications to be sent to that phone, the app must send a token to identify the device. This is done via the */users/user/device-token* endpoint. This device token is associated to the user and saved on the server. This is the token used to send notifications via Apple or Google services.

The workflow to send a notification is depicted in Fig. 4:

1. An application sends a message to the Notify server (the message is linked to a service).
2. The server sends a notification to both Apple Push Notification service [10] and Firebase Cloud Messaging [11] (depending on the device token type) for all users who subscribed to that service.
3. The notification is sent by Apple and Google cloud services to the user device.
4. When the user opens the notification, the full message is read from the Notify server API.

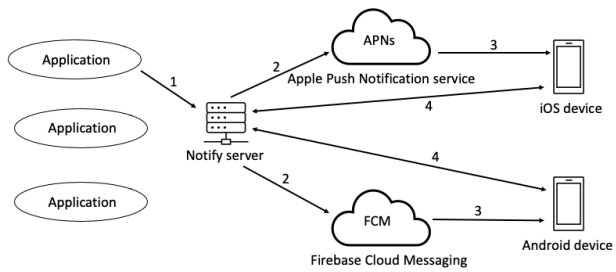


Figure 4: Notification workflow

The Notify server is deployed using docker in a Virtual Machine at ESS and on Kubernetes at MAX IV. All site specific parameters can be defined in a configuration file.

APPLE IOS CLIENT

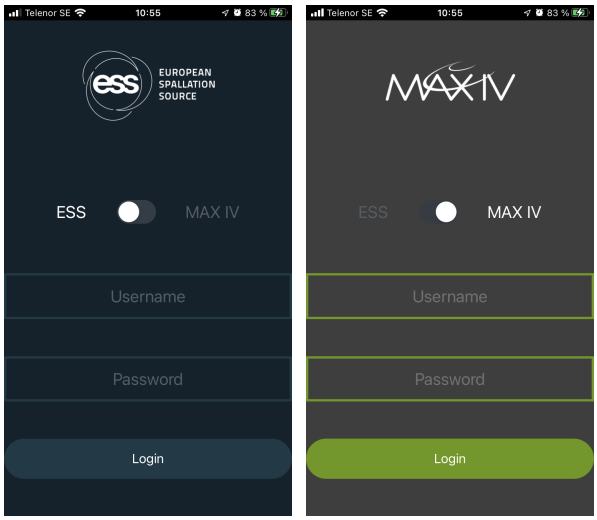
The iOS client is a GUI for the REST API described previously. Once installed it will contact the server through a link that is hard coded in the application. In our current version we support two possible links and customizations, one for ESS server and one for MAX IV server as in Fig. 5. As described previously, upon successful login a JSON Web Token is stored in the app and will be used for future authentications.

The app will also ask the user to accept notifications and in case of an affirmative answer it will request to Apple a push token that is sent to the Notify server. This token is stored in the profile of the user and used to push any new notification to the phone through the Apple Push notification API.

Once the app is properly registered on the server the available services can be selected for subscription (Fig. 6). The app will communicate to the server which service the user wants to subscribe to and the server will associate the push token of the user to the service. In this way every time a new message is delivered to a service, all the subscribers will be notified about the new message (Fig. 7).

When the app is opened, the main screen will display the list of available notifications (Fig. 8a), these are downloaded from the server at every startup of the app or if a new notification arrives and the app is running on the phone. The notifications are marked with a red dot if unread, and a badge

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV011



(a) ESS (b) MAX IV

Figure 5: Login Screen

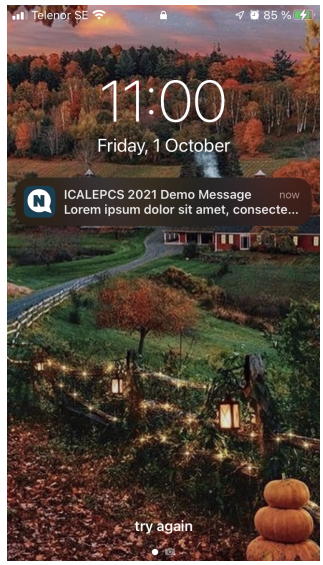


Figure 7: Push notification alert

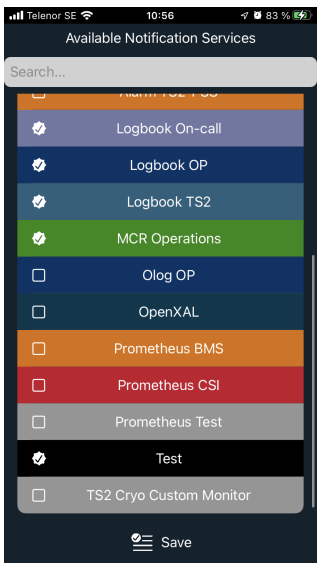
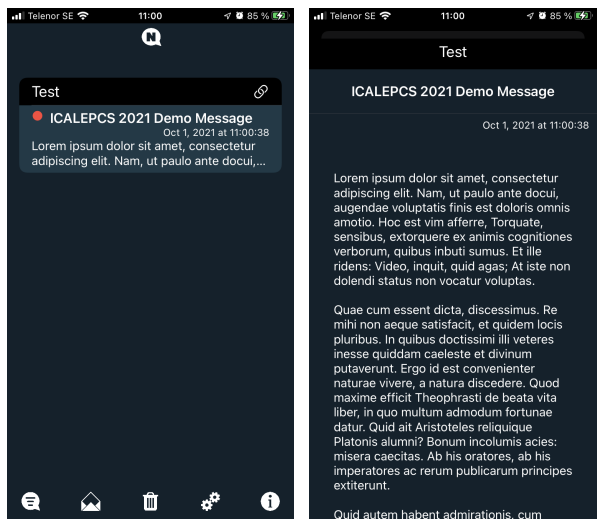


Figure 6: Services



(a) Messages list (b) Single message view

Figure 8: Main Screen

with the number of unread notifications is shown on the icon of the app. The top bar of the notification contains the name and the color of the service that is notifying, and on the right corner there is a button with a link to the URL sent together with the message, if present.

Tapping on a notification is possible to expand the text and read the full content (Fig. 8b).

Deployment

To be able to send push notifications to an Apple device the developer needs to request a push key that has to be used in order to send the notifications to the app. This is the reason why we developed a unique app with two possible logins (ESS and MAX IV), because in this way we can use the push key registered under the ESS license to push messages also from MAX IV. The alternative is that MAX IV register its own developer on the developer program of Apple and requests a key for push notifications that is set in the Notify server settings. This is under consideration for the future.

Apple offers multiple way to deploy applications. The most common way is to use the App Store, but our application has an access restriction to the users of ESS and MAX IV and Apple forbids the distribution of restricted applications on the App Store. The solution that we adopted to distribute the app is then through the Mobile Device Management (MDM) system. The whole process is exactly like to distribute the app through the App Store, it means that the app must be signed with the distribution signature generated on the Apple Developer portal and sent to the App Store Connect, that is the website that Apple uses to manage the distribution of apps. Once the app is on the portal and all the relevant information for distribution are filled, the app is

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV011

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

sent to Apple for review as Private. A normal app in general is sent as public, but because we do not go to the App Store but we will stay within our MDM, the app has to be marked as private.

Apple will then verify the application running it, this means that together with the app they will require a demo account to test the functionalities, so we created in our LDAP server a test account that we enable every time we submit a new revision of the app for approval. After the checks of Apple the application is marked as Ready for Sale and appears automatically to the registered Business web page connected to our license for distribution. For ESS we use the Self Service application to distribute internal apps.

This process, except the push key that is always the same, has to be repeated for each version of the app submitted to Apple.

ANDROID CLIENT

The Android client was initially developed after the iOS counterpart, consequently the functionality and user interface (UI) were copied to be basically identical. Some minor difference in the UI layout exist, though. Figure 9 shows the lock screen showing a test notification message.

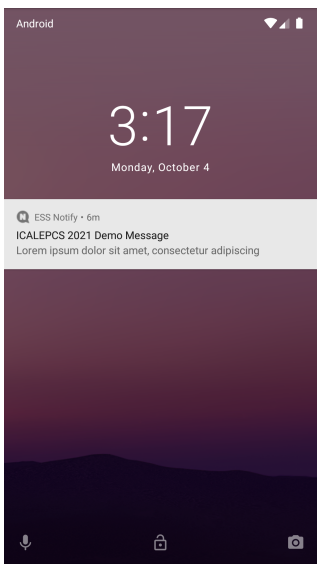


Figure 9: Push notification alert

The Android client does not support selection between ESS and MAX IV, it currently only supports the ESS use case. To be used at MAX IV, the application was recompiled after changing the Notify server url. This was enough for testing purpose. More customization is planned to make the theme MAX IV specific.

The current code base of the Android client requires a minimum version of Android 7.0, which was released August 2016.

Deployment

Development of Android apps is free of charge, as are development tools. Installing an Android app by downloading an apk (Android application PacKage) file to any compatible target device - or using USB tethering - is possible and does not incur any limitations to the app usage. ESS has however opted for distribution over Google Play due to its simplicity. The Google Play approval process is slightly simpler compared to the process used by Apple, e.g. there is no test account needed in order for Google to be able to login to the application. This distribution will be considered by MAX IV when the app is customized.

In order to publish apps on Google Play one needs to register a developer account, currently priced at $25 (one-time charge). The developer account must be linked to a Gmail account.

APPLICATION INTEGRATION

As we saw earlier, sending a notification only requires a POST request to the Notify server. ESS integrated this system in several applications: LogBook, OpenXAL, EPICS alarms and Prometheus are some examples. Integration can be done in different ways depending on the application. For Prometheus, a bridge [12] was created to forward and curate messages from the Alert manager to the Notify server. EPICS alarms are sent via Kafka, which makes it quite flexible. A client is used to listen to the desired Kafka topics and forward alarms to the server. At MAX IV, preliminary tests were performed successfully with Achtung, a new alarm management system for Tango.

CONCLUSION

The solution put in place at ESS relies on modern tools and native mobile applications for both iOS and Android. The management of who will receive notifications is delegated to the users themselves, which removes the need for a central configuration that can be fastidious to keep up to date. This is really beneficial to the users who can subscribe and unsubscribe as they want, from their phone. It gives an unified way to receive notifications from different systems. The implementation is generic and can be re-used. The Notify server was straightforward to deploy at MAX IV and the concept was tested successfully. The customization of the mobile clients requires a bit more work but nothing major.

REFERENCES

[1] E. Laface. (2021) Notify iOS Client. https://gitlab.esss.lu.se/ics-software/ess-notify

[2] G. Weiss. (2021) Notify Android Client. https://gitlab.esss.lu.se/ics-software/ess-notify-android

[3] B. Bertrand and E. Laface. (2021) Notify Server. https://github.com/EuropeanSpallationSource/notify-server

[4] S. Ramírez. FastAPI. https://fastapi.tiangolo.com

[5] T. Christie. Starlette. https://www.starlette.io

[6] S. Colvin. pydantic. https://pydantic-docs.helpmanual.io

[7] PostgreSQL. https://www.postgresql.org

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV011

[8] OpenAPI standard. https://github.com/OAI/OpenAPI-Specification

[9] Swagger UI. https://github.com/swagger-api/swagger-ui

[10] Apple Push Notification service. https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/sending_notification_requests_to_apns/

[11] Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging

[12] A. Curri. (2021) Alertmanager To Ess Notify. https://gitlab.esss.lu.se/ics-infrastructure/alertmanager-to-ess-notify

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

LHC COLLIMATION CONTROLS SYSTEM FOR RUN III OPERATION

G. Azzopardi*,[1], S. Redaelli[1], G. Valentino[2], B. Salvachua[1], M. Solfaroli Camillocci[1], M. Di Castro[1]
[1] CERN, Geneva, Switzerland
[2] University of Malta, Malta

Abstract

The Large Hadron Collider (LHC) collimation system is designed to protect the machine against unavoidable beam losses. The collimation system for the LHC Run 3, starting in 2022, consists of more than 100 movable collimators located along the 27 km long ring and in the transfer lines. The cleaning performance and machine protection role of the system critically depend on the accurate positioning of the collimator jaws. The collimation control system in place enables remote control and appropriate diagnostics of the relevant parameters. This ensures that the collimators dynamically follow optimum settings in all phases of the LHC operational cycle. In this paper, an overview of the top-level software tools available for collimation control from the control room is given. These tools range from collimator alignment applications to generation tools for collimator settings, as well as collimator scans, settings checks and machine protection sequences. Amongst these tools the key upgrades and newly introduced tools for the Run 3 are presented.

INTRODUCTION

The CERN Large Hadron Collider (LHC) accelerates and collides two counter-rotating beams towards the unprecedented design centre-of-mass energy of 14 TeV [1]. It is made up of eight arcs containing superconducting magnets and eight straight sections that are referred to as insertion regions (IRs) [2].

The quench limit of the LHC superconducting magnets is of the order of 10–30 mW/cm^3 [3] and the damage limit of metal is of a few hundred kJ per cm^3 [4], to be compared to the stored beam energy of more than 300 MJ planned for Run 3. Therefore, a high-performance and robust collimation system is installed to safely dispose of beam losses in the collimation regions, providing a cleaning efficiency of 99.998% of all halo particles [5].

A total of 123 movable collimators are installed in the LHC for the start of Run 3 in 2022, whereby 14 betatron collimators, 6 injection protection collimators and all 12 transfer line collimators have been newly installed/replaced and 2 crystal collimators are planned to be replaced. The LHC ring collimators are mainly concentrated in two dedicated cleaning insertion regions (as shown in Figure 1); IR7 is dedicated to the betatron cleaning, and IR3, where the dispersion is larger, provides off-momentum cleaning [6]. Other collimators are installed in the experimental region to protect the inner triplet magnets, and in the high-luminosity regions IR1 and IR5, to dispose of the collision debris.

* gabriella.azzopardi@cern.ch

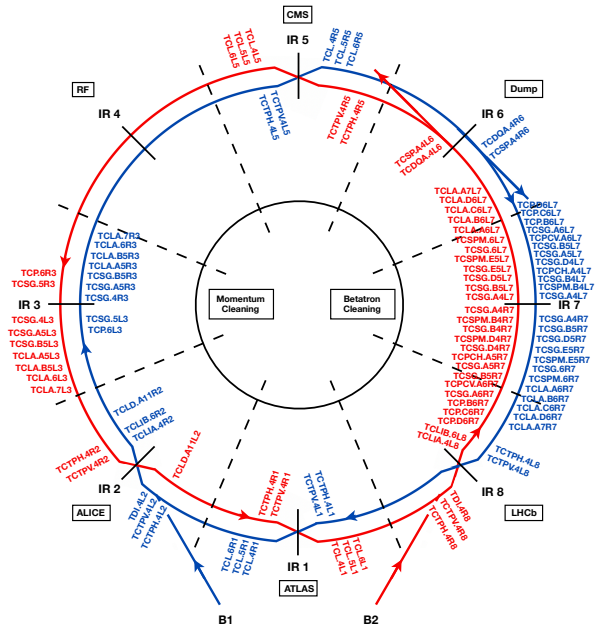


Figure 1: The Run 3 LHC ring collimation system layout.

The collimators have different designs, orientations (horizontal, vertical or skew) and roles for cleaning and protection (refer to Table 1); (1) Primary, secondary collimators, and shower absorbers are located in IR3 and IR7, (2) Tertiary collimators protect the super-conducting triplet quadrupoles in all experimental regions, (3) Injection protection devices (TDI, TCLI) protect the machine in case of injection errors (also the transfer line (TCDIL) collimators), (4) Dump collimators protect against asynchronous or unclean beam dumps in IR6, (5) Crystal collimators, in IR7, enhance ion beam cleaning, (6) Fixed aperture, passive collimators in IR3/7 shield specific magnets from high radiation doses (TCAP).

Table 1: Movable LHC Ring Collimators

Collimator	Description	Number
TCP	Primary	8
TCSG/TCSP/TCSPM	Secondary	28/2/9
TCT	Tertiary	16
TCLA	Shower Absorber	18
TCL	Physics debris	12
TCLD	Disp. Suppressor	2
TDI/TCLI	Injection Protection	6/4
TCDQ	Dump Protection	2
TCPC	Crystal collimator	4

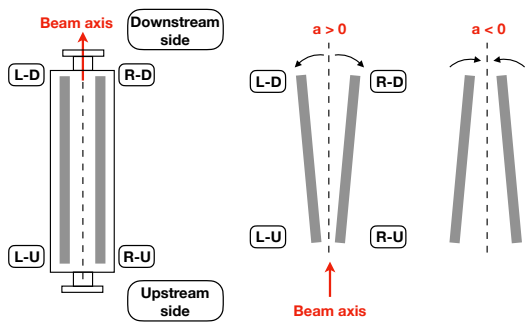
Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

BACKGROUND

Collimator Design and Control

A collimator is made up of two parallel absorbing blocks, referred to as *left* and *right* jaws, which are typically positioned symmetrically around the beam, except for the 2 one-sided TCDQ. Each jaw is controlled by two stepping motors to precisely adjust the jaw position and angle with respect to the beam [7]. The maximum and minimum possible angles are ±2 mrad [8]. The collimation coordinate system is displayed in Fig. 2, whereby the jaw corners are referred to as left-up (LU) and right-up (RU) when they are upstream of the beam and left-down (LD) and right-down (RD) when they are downstream of the beam (or at the end of the beam).



(a) Jaw coordinate system (b) Jaw angular tilt convention

Figure 2: (a) The collimator coordinate system and (b) the jaw tilt angular convention as viewed from above, from [9].

Each collimator has Four Linear Variable Differential Transformer (LVDT) sensors and resolvers to independently measure the position of the collimators' axes in real-time at 100 Hz [10]. An additional 2 LVDTs are installed for direct collimator gap measurements, resulting in a total of 6 LVDTs per collimator used for jaw position interlocking [11, 12]. Moreover, the tanks housing 60 of the collimators have an additional motor, resolver and non-interlocked LVDT, to move the entire collimator in the orthogonal plane. This feature allows exposing a fresh surface to the beam in case of suspected collimator damage from beam losses. Finally, 33 collimators have 2 Beam Position Monitoring (BPM) pick-up button installed in each jaw, to provide a direct measurement of the beam orbit at the collimator location [13]. A third BPM is mounted on the collimator vacuum tank of 13 of these collimators, to measure the beam position on the axis orthogonal to the cleaning plane.

The tightest settings of 5 σ for the TCPs correspond to gaps of ~ 1 mm at 7 TeV, requiring ~ 20 µm position accuracy. To ensure that, in static and dynamic conditions, the jaw positions stay within safe operational windows, a complex system of threshold functions has been implemented [7]. The limits are defined as functions of time, beam energy, and β^* functions that express the amplitude of the beam oscillation in the IP (or beam size) [4].

Four limit functions (inner and outer dump and warning limits) can be defined for each LVDT, for a total of 24 functions. If the measured axis position violates any of the dump limits, which are always active in parallel, the low level control system requests an immediate abort of the circulating beams. The interlocks per collimator for jaw position and gap measurements include; limits versus time (inner and outer thresholds for 6 degrees of freedom, i.e. 12 limits), maximum allowed gaps versus beam energy (2 limits) and gap thresholds versus β^* (inner and outer thresholds for 2 gaps, i.e. 4 limits). The energy and β^* dependent limit functions catch failures in triggering the start of time functions at the start of energy ramp: a beam dump is eventually requested if the collimator gaps are not scaled down as needed by smaller beams [7]. The list of key parameters for the collimation system is presented in Table 2 for a typical LHC operational cycle that includes separate functions for the energy ramp, betatron squeeze and collision process. Injection protection collimators require a simple implementation, as they do not need to be controlled through functions of time.

Table 2: Collimation System Parameters (excluding TCDQ)

Parameters	Number
Movable collimators in the ring	109
Transfer line collimators	12
BPM pick-ups	145
Stepping motors	544
Resolvers	536
Position/gap measurements	786
Interlocked position sensors	726
Motor settings Vs Time	1452
Threshold settings Vs Energy	242
Threshold settings Vs β^*	484

Collimation Controls Software Technologies

The collimation controls are used for all LHC ring and transfer line collimators. The controls consist of various software applications, each one providing a different set of user functionality, ranging from; collimator alignments and settings' controls to loss map and machine-protection check tools.

Various programming languages and tools are involved in developing the controls, including; Java for user interfaces, Front-end software architecture (FESA) [14] for device control, Python and SWAN [15] for machine learning and data analysis, LHC Software Architecture (LSA) [16] with Oracle for operational settings management, Next CERN Accelerator Logging Service (NXCALS) [17] for data logging and Apache Spark for data processing.

COLLIMATOR ALIGNMENTS

At the start of each year, the LHC goes through a commissioning phase to ensure that all collimators are correctly set up and ready for nominal operation [18]. During the commissioning phase various alignment campaigns [19] take

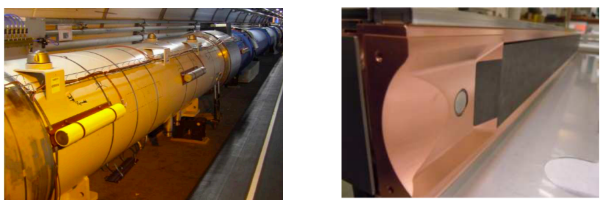
Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

place at different machine states, to set up the correct collimation hierarchy. Collimator alignments are performed with "setup beams" i.e. low intensity, which at the LHC typically corresponds to 3 bunches at 10^{11} p/beam. The collimator settings are monitored along the year as the beam orbit may shift over time [20], thus potentially requiring the collimators to be realigned. Moreover, different collimator setups are required when the machine parameters are changed.

Two types of beam instrumentation are available to align collimators (refer to Fig. 3); the Beam Loss Monitoring (BLM) and the in-jaw collimator Beam Position Monitoring systems. Each collimator has a dedicated BLM installed outside the beam vacuum, immediately downstream. BLMs are used to detect beam losses generated when halo particles impact the collimator jaws, such that the recorded losses are proportional to the amount of intercepted beam, measured in units of Gy/s. Meanwhile, BPMs directly embedded in the collimator jaws, allow for a faster alignment by analysing the electrode signals, without needing to touch the beam [13].



(a) BLM detector (b) BPM pick-up button

Figure 3: (a) BLM attached to LHC cryostat (yellow device) [21] and (b) BPM pick-up button embedded in collimator jaw [22].

BPM Alignment Controls

BPM collimators are aligned using a successive approximation algorithm. This takes into account a calibration of the nonlinear BPM sensitivity to beam displacement and an asymmetry of the electronic channels processing the BPM electrode signals [23].

The BPM controls allow users to monitor the electrodes and the beam position. The user can automatically align collimators by defining the minimum collimator gap (mm), the target alignment error (µm) for alignment precision when equalizing the electrodes, and the time interval(s) between each step of the successive approximation algorithm.

The BPM-based alignment procedure aims at finding the jaw positions and angles, where the beam orbit is centred, at both the upstream and downstream sides of the collimators, individually. The algorithm works by moving both collimator jaws in steps, keeping the same gap, until the signals from electrodes are equalized as shown in Fig. 4 [22].

The entire procedure is automated, able to align a single collimator at the optimal angle in ~10–20 s. In addition, all BPM alignments can be performed in parallel [13]. The method is not invasive and does not cause any beam loss.

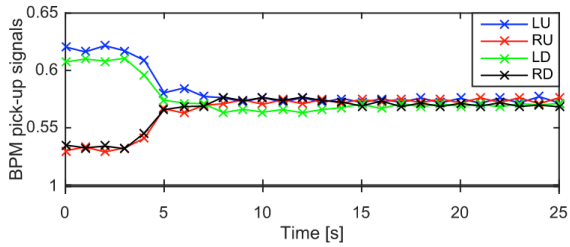


Figure 4: BPM electrode signals during a BPM-based alignment in 2017, from [13].

BLM Alignment Controls

All collimators can be aligned using the beam loss signals recorded by their respective BLM located directly downstream. A beam-based alignment (BBA) procedure [24] is used to align collimators with BLMs. This involves creating a reference halo with the primary collimators, then moving the collimator jaws separately towards the beam in steps of 5–20 µm. A collimator is said to be aligned when both jaws are centred around the beam after touching the beam halo. The alignment of any collimator relies on BLM signal distinction between alignment and spurious spikes: a collimator must continuously move towards the beam ignoring any spurious spikes, until a clear alignment spike is observed. Throughout the procedure, the jaws are kept parallel to each other with a zero tilt angle with respect to the collimator frame. The procedure can be extended to measure the jaw angle with respect to the beam [19].

The BLM controls allow users to monitor the BLM signals and the collimator position. Since 2018, the entire BBA procedure has been automated [25, 26] by introducing supervised machine learning to classify alignment spikes [27] and other algorithms [28]. The user can start an automatic alignment by selecting the list of collimators to be aligned and the software will align all collimators and save the settings.

This new automatic software is able to align a single collimator in ~1–2 mins [29, 30], depending on the machine state. When aligning collimators with BLMs, a maximum of two collimators can be aligned in parallel, one from each beam. However, the two collimators must be specifically selected, as one must consider possible cross-talk, i.e. having multiple BLMs around the ring detecting the losses generated by a single collimator [31].

Having the foundations for an automatic BBA opens the possibility for further research and development [32]. It also allows introducing more complex alignment techniques and changing the collimation hierarchy more frequently throughout an operational year, to provide tighter settings.

Angular Alignment The current operational settings for the betatron cleaning hierarchy requires a 1.5 σ retraction between primary and secondary collimators in IR7, corresponding to ~ 300 µm. Tighter collimator settings with smaller retractions might be used in the future to improve the LHC performance and achieve a smaller β^* [33]. Keep-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

ing collimators with a zero tilt angle with respect to the beam will not be adequate to operate the system with retractions below 1.5 σ [34]. Moreover, possible tank misalignments are a source of error that could jeopardize the performance of the system if not corrected. Aligning collimators at an angle will allow for tighter collimator settings. However, this procedure is longer, as it involves performing the standard BBA at different angles to find the most optimal one.

The angular alignment controls allow users to monitor the BLM signals and the positions of the 4 collimator jaw corners. Angular alignments are implemented on top of the BBA, therefore they have also been automated in 2018. The user can automatically align collimators at an angle by setting the start and end angles (µrad) to define the angular range to be explored, and the angle step size (µrad) to tilt the collimator between each alignment. The user can then select one of the 3 angular alignment procedures developed in [35], depending on the specific error encountered; (1) The first method aligns the collimator at different angles (keeping the jaws parallel). This is used to determine the optimal angle for the collimator tank in cases where there is an offset. (2) The second method aligns the upstream and downstream jaw corners individually, to determine the corresponding centres. The optimal angle is taken to be the average of the two. (3) The third method individually aligns the collimator jaws at different angles and aligns the other jaw (with zero tilt) to act as a reference. This method is useful in cases of asymmetries within the collimator itself, as the optimal angles for both jaws are determined independently.

At injection, methods 1 and 3 require a similar amount of time ~13 minutes, whilst method 2 requires ~3 minutes [36]. Since angular alignments rely on BLMs, one must consider possible cross-talk when aligning collimators in parallel.

LOSS MAP ANALYSIS FRAMEWORK

The collimation system performance is monitored when the configuration is changed, and at most every three months of operation, as the beam orbit may shift over time. This is done by inducing slow (multi-turn) beam losses, such that a large number of particles are purposely sent onto primary collimators in a controlled way, and the resulting electromagnetic showers can be detected by BLM devices around the LHC ring. A beam loss map, showing the spatial distribution of the measured losses along the LHC ring (Fig. 5), can then be generated to validate the collimator setup [36]. Specifically the losses in IR7 provide information on the collimation hierarchy and halo cleaning performance. In case of a degradation in performance, the collimators might need to be re-aligned to generate a new setup for the current beam state. This could be observed as a degradation of the cleaning in the dispersion suppressor of IR7, or a difference in the pattern of the beam losses at the collimators [37] or by the observation of anomalous loss locations.

A new framework was developed for Run 3, providing an automatic way to validate loss maps generated throughout the year. It allows users to import loss map metadata

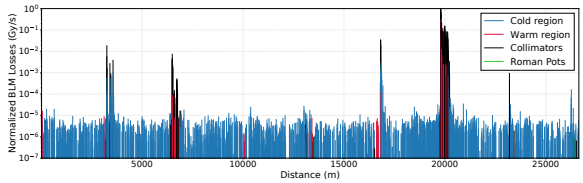


Figure 5: Proton full ring loss map performed at injection in the horizontal plane for beam 1, from [36].

as new loss maps are generated, including; start date/time, plane info, beam energy, and loss map type (betatron or off-momentum). This metadata is then used to automatically extract the loss map data logged in NXCALS. The user then has the option to plot individual loss maps depicting the distribution of losses across the full ring or in specific IRs. Alternatively the user can plot multiple loss maps at a particular IR, stacked for comparison. This framework also integrates the necessary tools for automatically calculating the collimation hierarchy inefficiency by analysing the losses in the dispersion suppressor.

SETTINGS GENERATION APPLICATION

Once the collimation hierarchy is validated, the next step is to generate the collimator settings for operation [38]. As previously mentioned, the collimator settings obtained following the BBA are saved to file. A new application was developed for Run 3 to automatically extract the relevant data and calculate and import these settings into the control system defined in LSA. A mathematical framework for defining these functions, which will now be automatically calculated by the system, can be found in [38]. This application replaces the scripts that were used in the past, creating a central and homogeneous procedure for settings' generation.

The user can automatically generate the settings by selecting the BBA results file, the list of collimators, the LSA parameters to update and the collimator settings to be used. In addition, the user can manually edit any of the settings loaded into the application and save the modified settings to a new file for future use.

SETTINGS CHECK APPLICATION

Following their generation in LSA, the settings must be validated to ensure they are consistent with the expected collimator positions. The setting-check application requires the user to select the LSA configuration, the start and end time, and the list of collimators to check their settings. This application automatically compares the settings defined in the following locations: 1) the selected LSA configuration, 2) the settings calculated by the application based on data stored in LSA, 3) the data logged in NXCALS for the selected time period, 4) the real-time positions of the collimators.

MACHINE PROTECTION SEQUENCE

The precise control of the jaw positions versus time and beam position has important relevance for machine protec-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

tion (MP) of the LHC. As part of MP procedures, the collimation MP sequence, independently of the beam, checks that for every scenario, the interlocks are correctly triggered (recall Table 2). MP tests ensure the correct behaviour of interlocks to guarantee a beam abort in case of potentially critical situations, i.e. if a collimator moves beyond the safe limit [39]. A dedicated MP sequence for LHC collimators forms part of the collimation controls. It is an application which automatically tests all interlocks associated with a collimator. This application is used when new collimators are installed or replaced, and, after each long shutdown in order to test all collimators before starting a new LHC run.

To begin the testing sequence, the user simply selects a collimator. The collimator is moved sequentially to each of its limits to ensure the interlocks are correctly triggered, whilst the user monitors the results. This typically takes less than 10 minutes per collimator. Multiple sequences can be started in parallel for collimators that are not connected to the same interlock units (typically one interlock unit per IR).

BPM-CALIBRATION APPLICATION

This application is used to measure BPM non-linearities, by characterizing the BPM readout for selected jaw gaps and beam offsets, in order to validate the BPM calibration. This is achieved by performing an automatic scan which measures the beam position at different gaps and different collimator offsets, to ensure the centre is proportionate [23].

Before starting the scan the user must first align the collimator using the feedback from the BLM, to centre the jaws around the beam. To begin a scan, the user must define; the maximum offset (µm) to avoid beam scraping, the jaw step size (µm), the gap step size (µm), the initial and final gaps (µm) defining the range to be explored, and the waiting time between each movement(s).

The scan begins by opening the collimator jaws to the starting gap. The jaws are then continuously shifted by the predefined jaw step size, waiting the requested time between movements. When one of the jaws reaches the predefined limit, the jaws are moved back to the starting point and the jaw gap is reduced by the predefined gap step size. The procedure is repeated until the minimum jaw gap is reached. The jaw positions and BPM electrode signals are automatically saved to a file for offline analysis.

CRYSTAL COLLIMATION CONTROLS

The crystal collimation scheme uses bent crystals as primary collimators. These are used to channel beam halo particles and deflect them onto a single secondary collimator per beam and plane, further downstream, as opposed to the present multi-stage collimation system. The crystal-based collimation system improves the overall cleaning inefficiency in the IR7 dispersion suppressor and along the ring, with respect to the standard collimation system [40], for heavy-ion beams. In order to determine the optimal channelling angle, angular scans are performed by changing the crystals' orientation angle with respect to the beam envelope. Crystal channelling is achieved when a reduction in the local beam losses is observed in the corresponding BLMs [41].

The crystal controls allow users to control the goniometer that houses the crystal, whilst monitoring the signals of neighbouring BLMs and the linear crystal position. The user can align the crystal linearly using the BBA similar to one-axis collimators, and can perform an angular scan of the crystal to determine the optimal channelling angle for operational use [42].

Linearly aligning each crystal requires ~5–10 minutes depending on the machine state, as the automatic alignment is not yet available for crystals. Moreover, as this alignment relies on BLMs, one must consider possible cross-talk when aligning crystals in parallel. Scanning the largest angular range available of 20 mrad can take longer than an hour. Angular scans cannot be done in parallel, as any cross-talk would make it difficult to determine the optimal channelling angle.

GLOBAL MONITORING DISPLAYS

Given the numerous collimators, various global monitoring displays are available to provide an overview of their status. Four dedicated fixed displays are available for the collimators in beam 1, in beam 2, in the transfer lines and for crystals. An additional display for collimator settings is available to monitor the collimation hierarchy in case of incorrect settings. Finally, the number of BPM collimators have doubled for the start of Run 3. Therefore, a dedicated display for them is being developed to provide an overall status, comparing the collimator centre and the beam centre measured by the BPMs in real-time.

CONCLUSION

The LHC makes use of a high-performance collimation system to protect its sensitive equipment from unavoidable beam losses. A total of 123 movable collimators are readily installed in the LHC for the start of the Run 3 in 2022. This paper presents the control system of the LHC collimators that provide the functionality required to handle high-intensity beams. The system underwent a significant upgrade to improve the functionality in light of the recent system upgrade for Run 3. An overview is given of the software tools available for collimation, including; BPM alignments, BLM alignments, BLM angluar alignments, loss map analysis, generation of settings, machine protection sequence, BPM calibration, crystal collimation and global monitoring. These tools ensure precise jaw positioning during all the phases of LHC operation. The occurrence of unsafe conditions is minimized by making sure that all the critical degrees of freedom stay within safe operational windows. All these tools are readily available for the start of Run 3.

ACKNOWLEDGEMENTS

This work is presented on behalf of the LHC collimation team (BE-ABP-NDC), with special thanks to the LHC OP crew (BE-OP) and the collimation HW team (BE-CEM).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV012

REFERENCES

[1] O. Brüning *et al.*, "LHC design report", vol.1, 2004.

[2] L. Evans, "Beam physics at LHC", in *Proc. of PAC*, pp.19–23, 2003.

[3] B. Auchmann *et al.*, "Testing beam-induced quench levels of LHC superconducting magnets", in *Phys. Rev. Accel. Beams*, 18(6), 061002, 2015.

[4] S. Redaelli *et al.*, "LHC collimator controls for a safe LHC operation", in *Proc. of ICALEPCS*, pp.1104–1107, 2011.

[5] R. W. Assmann *et al.*, "Requirements for the LHC Collimation System", in *Proc. of EPAC*, pp.197–199, 2002.

[6] Oliver Brüning, "LHC Design Report: The LHC Main Ring", vol. 1, CERN, 2004.

[7] S. Redaelli *et al.*, "Final Implementation and Performance of the LHC Collimator Control System", in *Proc. of ICALEPCS*, paper WEPMU020, pp. 1104–1107, 2011.

[8] D. Missiaen *et al.*, "The alignment of the LHC", No. CERN-ATS-2009-117, 2009.

[9] S. Redaelli *et al.*, "Operational performance of the LHC collimation", in *Proc. of HB*, pp.395–399, 2010.

[10] R. Aßmann *et al.*, "The final collimation system for the LHC", in *Proc. of EPAC*, pp.986–988, 2006.

[11] B. Puccio *et al.*, "The CERN beam interlock system: principle and operational experience", No. CERN-ATS-2010-128, 2010.

[12] B. Todd *et al.*, "The architecture, design and realisation of the LHC beam interlock system", in *Proc. ICALEPCS*, 2005.

[13] G. Valentino *et al.*, "Final implementation, commissioning, and performance of embedded collimator beam position monitors in the Large Hadron Collider", in *Phys. Rev. Accel. Beams*, no.8, 2017.

[14] M. Arruat *et al.*, "Front-end software architecture", in *Proc. of ICALEPCS*, vol.7, pp.310, 2007.

[15] D. Piparo *et al.*, "SWAN: A service for interactive analysis in the cloud", in *Future Generation Computer Systems 78*, pp.1071–1078, 2018.

[16] G. Kruk *et al.*, "LHC Software Architecture (LSA)-Evolution Toward LHC Beam Commissioning", in *Proc. of ICALEPCS*, paper WOPA03, 2007.

[17] J. Wozniak and C. Roderick, "NXCALS-Architecture and Challenges of the Next CERN Accelerator Logging Service", in *Proc. of ICALEPCS*, paper WEPHA163, 2019.

[18] J. Wenninger, "Operation and Configuration of the LHC in Run 2", No. CERN-ACC-NOTE-2019-0007, 2019.

[19] G. Valentino *et al.*, "Semiautomatic beam-based LHC collimator alignment", in *Phys. Rev. Accel. Beam*, 15(5), 2012.

[20] R. J. Steinhagen, "LHC Beam Stability and Feedback Control-Orbit and Energy", Ph.D. thesis, RWTH Aachen U., 2007.

[21] M. Stockner, "Classification of the LHC BLM ionization chamber", in *Proc. of DIPAC*, pp.328–330, 2007.

[22] G. Valentino *et al.*, "First operational experience with embedded collimator BPMs in the LHC", in *Proc. of the IPAC*, paper WEPMW034, 2016.

[23] G. Valentino *et al.*, "Successive approximation algorithm for beam-position-monitor-based LHC collimator alignment", in *Phys. Rev. Accel. Beams*, 17(2), 2014.

[24] R. W. Aßmann *et al.*, "Expected performance and beam-based optimization of the LHC collimation system", in *Proc. of EPAC*, pp. 1825–1827, 2004.

[25] G. Azzopardi *et al.*, "Software Architecture for Automatic LHC Collimator Alignment using Machine Learning", in *Proc. of ICALEPCS*, paper MOCPL04, pp.78–85, 2019.

[26] G. Azzopardi, "The Automatic LHC Collimator Beam-Based Alignment Software Package", in *Proc. of ICALEPCS*, paper WEPV016, this conference, 2021.

[27] G. Azzopardi *et al.*, "Automatic spike detection in beam loss signals for LHC collimator alignment", in *NIMA-Section A*, vol.934, pp.10–18, 2019.

[28] G. Azzopardi *et al.*, "Automatic Beam Loss Threshold Selection for LHC Collimator Alignment", in *Proc. of ICALEPCS*, paper MOPHA010, pp.208–213, 2019.

[29] G. Azzopardi *et al.*, "Operational Results of LHC collimator alignment using machine learning", in *Proc. of IPAC*, 2019.

[30] G. Azzopardi *et al.*, "Operational Results on the Fully-Automatic LHC Collimator Alignment", in *Phys. Rev. Accel. Beams*, vol.22, pp.1208–1211, 2019.

[31] G. Azzopardi *et al.*, "Data-driven cross-talk modeling of beam losses in LHC collimators", in *Phys. Rev. Accel. Beams*, 22(8), 2019.

[32] G. Azzopardi *et al.*, "New Machine Learning Model Application for the Automatic LHC Collimator Beam-Based Alignment", in *Proc. of ICALEPCS*, paper THPV040, this conference, 2021.

[33] R. Bruce *et al.*, "Calculations of safe collimator settings and β^* at the CERN Large Hadron Collider", in *Phys. Rev. Accel. Beams*, 18(6), 2015.

[34] A. Mereghetti *et al.*, "β^*-Reach–IR7 Collimation Hierarchy Limit and Impedance", Technical report, CERN, 2016.

[35] G. Azzopardi *et al.*, "Automatic Angular Alignment of LHC Collimators", in *Proc. of ICALEPCS*, pp. 928–933, 2017.

[36] G. Azzopardi, "Automation of the LHC Collimator Beam-Based Alignment Procedure for Nominal Operation", Ph.D. thesis, Univeristy of Malta, 2019.

[37] N. Fuster Martinez *et al.*, "Run 2 collimation overview", in *Proc. of Evian Workshop*, 2019.

[38] R, Bruce *et al.*, "Principles for generation of time-dependent collimator settings during the LHC cycle", in *Proc. of IPAC*, pp.3753–3755, 2011.

[39] S. Redaelli *et al.*, "2011 modifications of the LHC collimator controls relevant for machine protection", LHC-OP-MPS-0016, EDMS doc. 1119832, 2011.

[40] D. Mirarchi *et al.*, "Design and implementation of a crystal collimation test stand at the Large Hadron Collider", in *The European Physical Journal C*, 77(6), pp.1–13, 2017.

[41] R. Rossi *et al.*, "Status of crystal collimation studies at the LHC", in *Proc. of IPAC*, pp.84–87, 2017.

[42] R. Rossi, "Experimental assessment of crystal collimation at the large hadron collider", Ph.D. thesis, Uni. of Rome, 2018.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

WRAP – A WEB-BASED RAPID APPLICATION DEVELOPMENT FRAMEWORK FOR CERN'S CONTROLS INFRASTRUCTURE

E. Galatas*, A. Asko†, E. Matli‡, C. Roderick§, CERN, Geneva, Switzerland

Abstract

To ensure stable operation of CERN's accelerator complex, many Devices need to be controlled. To meet this need, over 500 custom Graphical User Interfaces (GUI) have been developed using Java Swing, Java FX, NetBeans, Eclipse SWT, etc. These represent a high maintenance cost, particularly considering the global evolution of the GUI technology landscape. The new Web-based Rapid Application Platform (WRAP) provides a centralized, zero-code, drag-n-drop means of GUI creation. It aims to replace a significant percentage of existing GUIs and ease new developments. Integration with the Controls Configuration Service (CCS) provides rich infrastructure metadata to support application configuration, whilst following the associated equipment lifecycle (e.g. renames, upgrades, dismantling). Leveraging the CERN Accelerator Logging Service (NXCALS) and the Unified Controls Acquisition and Processing (UCAP) platform, allows WRAP users to respectively, create GUIs showing historical data, and interface with complex data-stream processing. The plugin architecture will allow teams to further extend the tool as needed. This paper describes the WRAP architecture, design, status, and outlook.

INTRODUCTION

Over the past decade, a large number of expert applications have been developed to provide a way of controlling and monitoring thousands of Devices present within the CERN accelerator complex. The need for control and data visualization applies not only to production equipment, but is also essential for developing and testing new Devices. However, the ecosystem of technologies and platforms used for the development of such applications is quite fragmented. A lot of applications evolved organically based on individual needs. At the same time, the desktop graphical application tool kits traditionally used at CERN have evolved, whilst in parallel facing a major decline of community interest over the years. The Java Swing toolkit has been used since the late 90's, with Java FX coming on the scene several years ago. Most recently, given the direction of Oracle's support for Java as a graphical user interface solution, PyQt has been adopted for some graphical applications [1]. Unsurprisingly, the proliferation of custom Graphical User Interfaces (GUI) combined with an relatively rapid evolution of GUI technologies has lead to a worrying situation backed by extensive technical debt.

Instead of forcing a large and diverse community at CERN to continue learning new GUI technologies and develop their own applications, the following question was raised: "Can we turn the situation around and provide a central, data-driven GUI platform, which experts can use to configure their applications, based on their domain knowledge, without worrying about how to develop, build, deploy, and maintain it, using technologies which will undoubtedly continue to evolve?". A new Web-based Rapid Application Development Framework (WRAP) was designed as the answer to this question, and aims to replace many of the legacy applications over time. WRAP will provide a centralized solution around a common platform, integrated with core Controls sub-systems such as the Accelerator Logging Service (NXCALS) [2] and Controls Configuration Service (CCS) [3]. Since the inception of this project comes after years of application development in the organization, there is a large data-set of use-cases, feedback, and design decisions to analyse and improve upon. Leveraging this knowledge can shape WRAP into a single, unified solution for the vast majority of use-cases.

PROJECT GOALS

There are currently over 500 custom Graphical User Interfaces (GUI) relating to Device control and corresponding to different needs of Operation. Since WRAP encompasses many diverse use-cases and clients, it is important to have a clearly defined set of goals, based on which, decisions can be made regarding individual features.

The most vital attribute WRAP must exhibit, is ease of use. An intuitive User Interface (UI) is required, and beyond improving user productivity, it must encapsulate the complexity of Device modeling and communication. Wherever possible no-code, drag-and-drop configuration should be preferred, allowing experts without a programming background to work at a higher level. This level of abstraction must not, however, come at the cost of performance. Many parallel, real time visualizations should be supported.

Being a centralized service concentrates a lot of complexity on the platform itself, creating a barrier that may inhibit contributions from other teams. To offset this, contributions will instead be supported through a plugin architecture. A stable public application programming interface (API) will be made available giving access to the core functionalities of the platform. Those must in turn have high test coverage and be very conservatively altered.

Lastly, WRAP should leverage device metadata from the CCS to the highest possible degree. Not only does this remove redundant configuration steps, but also enables the restriction of configuration options to only those that are compatible with any given Device. This will help users to configure applications within WRAP whilst avoiding configuration compatibility errors that typically stem from a lack

* epameinondas.galatas@cern.ch
† anti.asko@cern.ch
‡ aemnuele.matli@cern.ch
§ chris.roderick@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

of detailed knowledge of the underling Device interfaces and behaviour.

ARCHITECTURE OVERVIEW

The WRAP architecture (Fig. 1) needs to accommodate the goals mentioned above, whilst also ensuring that it can scale in terms of functionality, complexity and performance. A clear separation between the rendering logic and underlying data management has to exist, while delegating some key points of complexity to relevant CERN Controls subsystems.

The platform's front-end (user interface layer), is created using web technologies rather than as a native desktop application. Running in a web browser removes cross-platform compatibility concerns, simplifies distribution and updates, and facilitates use from mobile devices. JavaScript's ever improving speed also removes the need for performance trade-offs in order to facilitate the WRAP platform use cases. Thanks to its immense popularity, the web stack appeals to a lot of potential candidates applying to CERN, it provides mature frameworks and UI design systems, and incorporates comprehensive debugging tools. A high degree of flexibility is also present; while the platform and most graphical widgets are being developed in the Angular framework, 3rd party widgets can utilize any preferred solution. These widgets can then be bundled and integrated as native Web Components. Web Components can encapsulate presentation and behaviour in a custom HTML element, creating the basis of a plugin architecture. Each widget has a specific, predefined structure that is modeled in the WRAP database. This model is used to validate widget states and to provide constraints on what types of data can be visualized.

In order to provide actual Device data to be rendered and processed, a variety of other Controls sub-systems are used. The information about the available Devices and their data structures is managed within the Controls Configuration Service (CCS) [3] from where it is served to WRAP via a REST API. After the desired Devices and their Properties are identified, the WRAP back-end subscribes to live updates of data values, using the CERN Java API for Parameter Control (JAPC) [4]. As values become available, they are aggregated and propagated from the WRAP server to corresponding WRAP clients via HTTP/2 push events.

Device Evolution

As operational needs or underlying hardware evolve, corresponding Controls software Devices follow. This can mean a simple change of Device name or a more complex change of Device interface. A requirement for WRAP is to be resilient to such events by design. This is achieved by integrating with the CCS and it's Controls Configuration Data Lifecycle (CCDL) manager [5]. Knowing the dependencies between Device data and WRAP widgets, throughout all applications configured within WRAP allows a WRAP API to be exposed to the CCDL. Using this API, the CCDL can interrogate WRAP to check for compatibility of CCS changes to Device configurations and, in most cases propagate the changes to WRAP - automatically keeping the applications working over time. Very complex changes such as data type alterations or Device deletions will at least result in notifications to the relevant application owners in order that they can make a manual intervention.

Historical Data

The architecture explained so far is able to visualize live Device data, however historical values are also important. For example, a graph widget would require data for the displayed time window to be pre-filled, before receiving updates based on current values. The visualisation of historical data is possible thanks to the integration with the CERN Accelerator Logging Service (NXCALS) [2], providing a unified solution for logging of time-series data acquired from accelerator Controls Devices.

Data Processing

For some use cases, Device data requires some form of processing before being displayed. This processing can range from simple filtering of specific values to aggregations that involve a large number of data subscriptions and require complex computations. The Unified Controls Acquisition and Processing (UCAP) [6] framework provides a means to facilitate and streamline acquisition and processing of Controls data. Data from multiple Devices are processed with pre-existing or user defined Python or Java code, with incoming values aggregated based on the desired event sourcing strategy. The result is then exposed as a Virtual Device, for which WRAP can utilize the standard infrastructure explained above to access Device metadata, live data and historical values.

Fault Tolerance

Given the foreseen critical nature of WRAP as a service (e.g. exposing critical GUI applications), it must have a high degree of fault tolerance. The WRAP front-end is received by the client as a Single Page Application (SPA). When loading a user application instance within WRAP, metadata has to be fetched in order to: render the included widgets, establish data source subscriptions, and fill historical values where necessary. Once initialized, WRAP applications only depend on the back-end infrastructure to exchange values, for which the corresponding server nodes are highly redundant. In the event of failure, such as a Device disconnection, wrong metadata, missing history, etc. parts of the application can fail gracefully, providing appropriate information to the user, without breaking the application as a whole.

CURRENT STATUS

WRAP development has now entered its second year. The processes for structuring and saving user applications have stabilized, and development is focused on adding editor features and extending the widget library. In this section, some

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

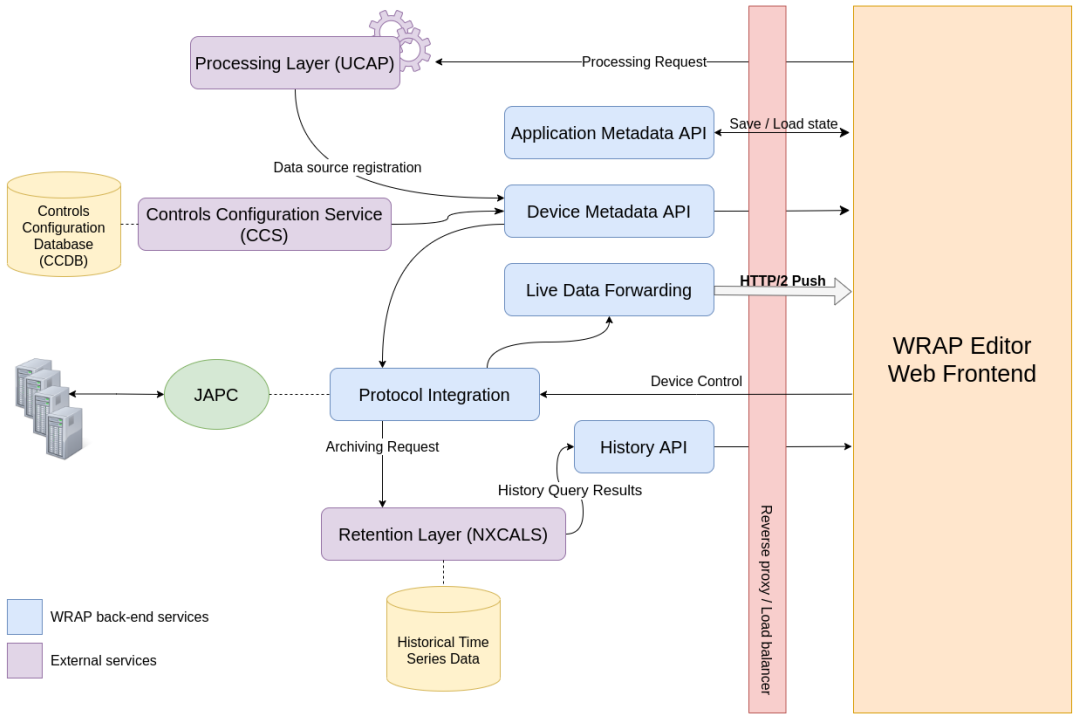


Figure 1: WRAP architecture overview.

of the existing capabilities will be highlighted coupled with some examples of finished products.

Figure 2 shows the WRAP editor. On the top, a tab-like list of recently visited applications is present. These tabs allow a fast navigation between applications. They also include information about the application title and owner, as well as an option to make a new copy - helpful when starting a new application based on an existing one. When an application owner enables the edit mode, a blue bar appears below with options to alter the application name and description, make it publicly visible, undo/redo functionality, and an option to enable a scroll-able / report-like view (instead of being limited to one screen).

On the left side is the Entry Point menu which allows the lookup of Devices (with some additional filters). Once a Device has been selected, a list of it's Properties appear, indicating the data type and additional flags dictating its behaviour. A Property value can either be used to instantiate a new widget that references it, or it can be assigned to an existing widget. In both cases, constraints are applied based on the correlation between the Property value type and corresponding widgets (e.g. boolean, string or enum can be shown on status indicators, numbers may be plotted on a chart, etc.).

Selecting a widget on the central application canvas will open a panel on the right side, from where the related attributes can be modified. Widgets can also be grouped together, with the resulting group having a union of the aspects of its children (Device source, colours, relative dimensions, etc.). This effectively enables bulk copying, moving and editing of related widgets. The bulk features around groups can make it very efficient to configure applications showing multiple aspects, of multiple Devices, of a similar type. For such cases, the user configures the individual widgets for a single Device, groups them together, copies them, then changes the Device binding in bulk for all widgets in the new group. Currently, the implemented widgets include graphs, labels, and status indicators, with many others foreseen.

ADOPTION

At CERN there are around 60 so-called "Fixed Displays" or "Vistars" which are applications that typically display important information about the status of accelerator operations and specific systems. These critical applications, are based on an aging custom framework, legacy Java Swing technology, and domain specific code. They are under the responsibility of a diverse group of people, who are only involved with software as an occasional, part-time activity, typically during periods of accelerator shutdown. It is not feasible for such people to learn the latest graphical technologies in their spare time. This is one of several reasons that it is foreseen to migrate all of these Fixed Display applications to the WRAP platform, thus lowering the technical debt and limiting the responsibility of the people associated with the applications to the domain knowledge and not the software maintenance.

A progressive migration of the Fixed Displays is already underway. Figure 3 shows one such application related to the LHC experiments. The applications created so far, aggregate data from multiple Devices via UCAP, and then visualize the resulting Virtual Device data from within WRAP. Currently, an effort is placed on evaluating how these new applications behave in the operational environment, while additionally

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

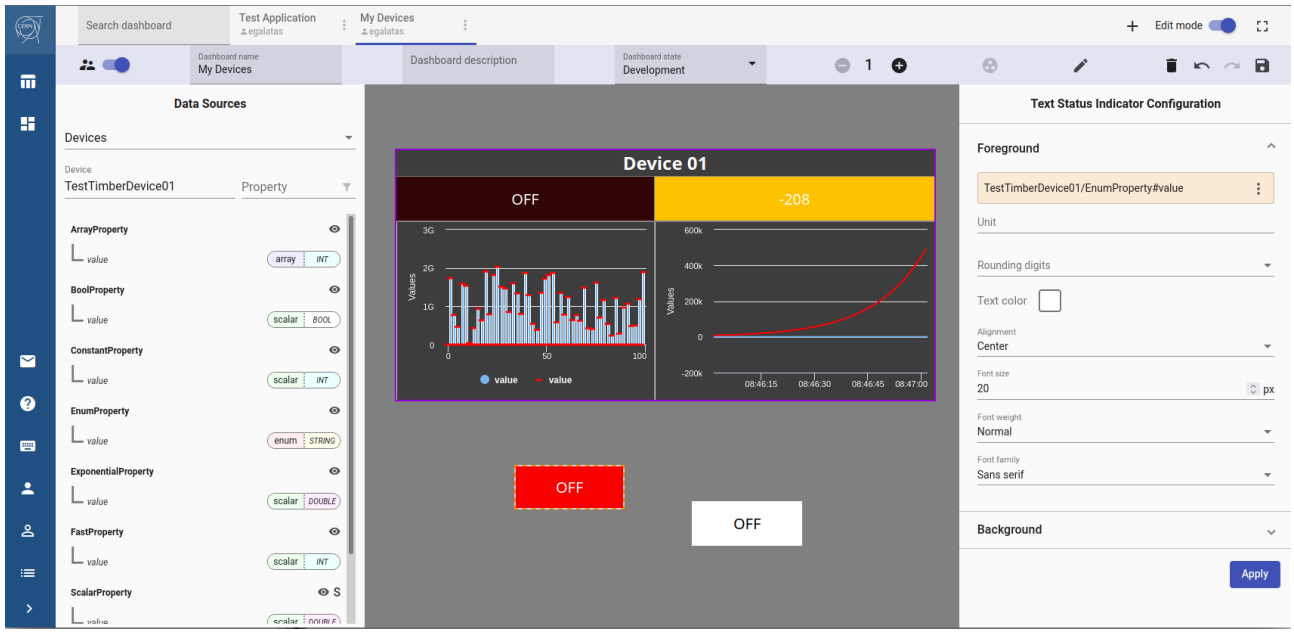


Figure 2: WRAP application builder.

evaluating possible cases of performance degradation or improvements with respect to the legacy ones. In order to give a desktop like experience, it is foreseen to evaluate technologies such as Electron [7], in order to avoid launching the WRAP applications via a full-blown web browser.

MAJOR CHALLENGES

As a system that combines many different services and provides solutions for very diverse use cases, WRAP presents a number of technical challenges that have to be addressed. Some of the most notable challenges currently experienced when developing WRAP are listed below.

Scalability and Performance

CERN's accelerator complex consists of hundreds of thousands of Devices, some of which emit very large data sets at very high frequencies. Creating visualizations that can cope with such cases requires optimizations both on data delivery and front-end (user interface) performance. Any inefficiency in either, however small, will be magnified and create bottlenecks in sufficiently complex user applications. A number of strategies have been employed to counter this, such as avoiding duplicate data subscriptions and bundling rendering events. Despite this, thorough regression testing will have to be incorporated for all widgets and major editor features in order to guarantee consistent performance across subsequent WRAP builds.

Evolving Technologies

Achieving stability on an evolving technological landscape is a challenge faced in all software. WRAP is particularly exposed to this problem, as it has to follow the evolution of not only 3rd-party front-end web frameworks and back-end technologies, but also the CERN Controls sub-systems with which it interacts. A consistent and well documented API between different parts of the platform, and a modular design across the entire stack, helps alleviate this problem. Even if breaking changes cannot be avoided, affected parts can be addressed separately, containing the scope of any potential refactoring and improving the accuracy of related cost estimations.

Web Limitations

As mentioned above, the web as an application run-time provides a number of advantages with respect to native applications, including efficient development times, easier debugging (especially around the visual elements), and a trivial distribution of the final product. There are however, some trade-offs that have to be considered.

The performance of complex applications on aging hardware can degrade the user experience. For WRAP this problem will become more apparent as more features are requested and more sophisticated applications are created. Solutions like Web Assembly and Web Workers have been considered and seem promising in eliminating these problems in the future.

Web applications might sometimes be limited due their inability to access native desktop features. They are also perceived differently by end-users, when faced with loading and running in a browser, instead of being installed as a standalone application available via a shortcut. However, all these point can be addressed by bundling WRAP with Electron [7] allowing the same behaviour and access to native APIs as a traditional desktop application. This also enables individual applications created within WRAP to be accessible via distinct shortcuts.

Finally, some problems are inherently more difficult to solve on the Web compared to native applications. For example, installation and communication across multiple windows, while possible, is a lot more involved. Creating multi-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

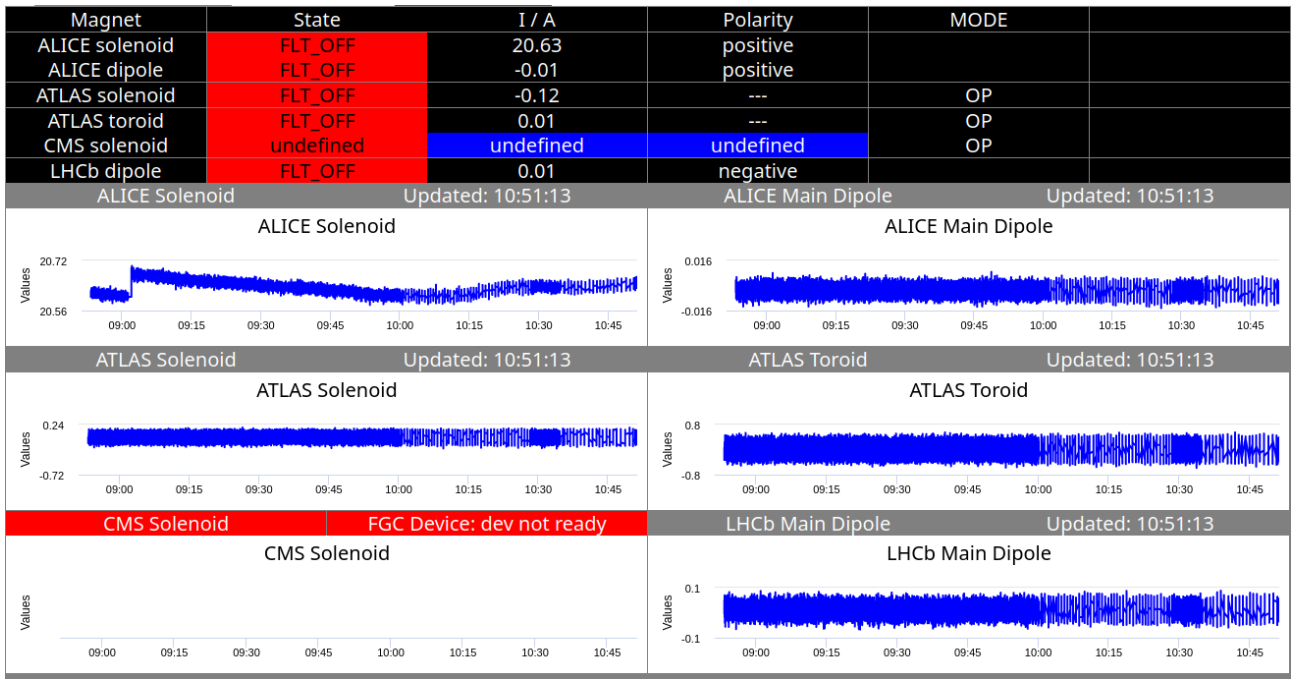


Figure 3: Example LHC Fixed Display implemented in WRAP.

windowed applications via WRAP is presenting a significant challenge. However, the easier development of most of the other parts of WRAP on the Web does, so far, greatly offset the small subset of features that prove problematic. Choosing any technology to build a sufficiently complex software will invariably highlight some deficiencies in some aspects of it.

NEXT STEPS AND FUTURE OUTLOOK

The aim is to use WRAP to replace as many as possible of the hundreds of CERN accelerator controls standalone graphical applications. From the outset, it is clear that WRAP, which is a data-driven application, will not be suited to replace the most complex legacy applications. Nevertheless, based on an in-depth analysis performed in 2020, it is expected that between 40-70 percent will be suited to WRAP. This will clearly require a lot of features to be present in the WRAP editor together with significant expansion to the widget collection. It is expected that the gradual adoption of WRAP by a large community will help establish collaborations with other teams (certainly inside CERN, and hopefully outside), which in turn can contribute improvements, extensions or even just valuable feedback to help WRAP become a more complete and useful end-product.

SUMMARY

WRAP introduces a new paradigm of developing applications for control and visualization of Device data. The new platform has received a lot of interest from the user community and, while development is at an early stage relative to the ambitious goals that where initially set, early adopters have shown promising results on the long-term viability of the solution. The overall architecture, consisting of modern and tested industry practices, coupled with the use of open-source software used by large communities, will help ensure the stability of the overall platform. WRAP is on-track towards it's objectives and it is expected that this endeavour will pay off, helping to substantially reduce technical debt, and minimising the need for diverse experts to acquire new software application development skills, in order to develop and maintain their graphical applications.

REFERENCES

[1] Z. Kovari *et al.*, "New Timing Sequencer Application in Python with Qt - Development Workflow and Lessons Learnt", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper THPV015, this conference.

[2] J. Wozniak *et al.*, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", in *Proc. ICALEPCS'19, New York, USA*, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[3] L. Burdzanowski *et al.*, "CERN Controls Configuration Service — A challenge in usability", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017. doi:10.18429/JACoW-ICALEPCS2017-TUBPL01

[4] V. Baggiolini *et al.*, "JAPC - the Java API for Parameter Control", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005.

[5] B. Urbaniec, "CERN Controls Configuration Service – Event-Based Processing of Controls Changes", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper MOPV043, this conference.

[6] L. Cseppentő *et al.*, "UCAP: A Framework for Accelerator Controls Data Processing at CERN", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper MOPV039, this conference.

[7] Electron, https://www.electronjs.org

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV014

ADOPTING PyQt FOR BEAM INSTRUMENTATION GUI DEVELOPMENT AT CERN

S. Zanzottera, S. Jensen, S. Jackson, CERN, Geneva, Switzerland

Abstract

As Java GUI toolkits become deprecated, the Beam Instrumentation (BI) group at CERN has investigated alternatives and selected PyQt as one of the suitable technologies for future GUIs, in accordance with the paper presented at ICALEPCS19.

This paper presents tools created, or adapted, to seamlessly integrate future PyQt GUI development alongside current Java oriented workflows and the controls environment. This includes (a) creating a project template and a GUI management tool to ease and standardize our development process, (b) rewriting our previously Java-centric Expert GUI Launcher to be language-agnostic and (c) porting a selection of operational GUIs from Java to PyQt, to test the feasibility of the development process and identify bottlenecks.

To conclude, the challenges we anticipate for the BI GUI developer community in adopting this new technology are also discussed.

INTRODUCTION

The software section of the Beam Instrumentation Group at CERN (SY-BI-SW) has a mandate, to provide expert GUIs allowing hardware experts to manage and diagnose instrumentation. As explained in detail in our preliminary evaluation[1], the software stack consists of several layers, where the most high-level ones, including the GUIs, have been traditionally implemented in Java. However, as Java GUI technologies age and become deprecated, efforts[1,2,3] were made to identify suitable, more modern replacements. As these evaluations concluded, no alternative Java framework could be identified, leading to the option of PyQt.

Adopting PyQt is not trivial: Firstly, tools, services and frameworks must be developed for integration with CERN's control system. Secondly, existing GUIs cannot simply be migrated – they must be rewritten. This represents a massive effort in terms of redesigning and reprogramming. In addition, developers will have to adopt Python as a programming language, which is a challenge in itself, as Python is fundamentally different from Java in many aspects.

As integration with CERN's control system is addressed by another team at CERN, we have been able to focus on the GUI programming aspect itself (GUI management tools, widgets, proof of concept GUIs) and also on the adaptation of our decade-old Java-oriented GUI development workflow to a more language-agnostic one, supported by more generic tools.

PYTHON TOOLS FOR EXPERT GUIS

Devtools: bipy-gui-manager

Previous integration efforts resulted in a set of tools and environments under the name of Acc-Py[4]. However, none of these tools are oriented towards GUI development: they all target a generalised Python codebase, favouring in practice CLI applications and libraries. Consequently, we proceeded to define a set of best-practices to be followed in order to develop Expert GUIs with PyQt and embarked on the creation of a specific tool, the "bipy-gui-manager" to encourage (and partially enforce) them.

Upon invocation, this command line utility collects some basic project information (project name, author name, author email etc…) and then creates a template project in the desired location, pre-configured with its own GitLab repository (created on the fly), a dedicated virtual environment, a template for Sphinx-based documentation and a workflow that provides Continuous Integration, Continuous Deployment and Continuous Documentation for the project. As a consequence, the setup effort required from the developer to get a fully standard project is close to zero. Even the README is pre-written by compiling a template README with the information gathered by the tool at setup time.

The bipy-gui-manager enables us to enforce group-specific conventions and promote best practices in general. This is valuable in homogenizing the code produced, given the number of short-term developers in the section and their different backgrounds.

One example is how bipy-gui-manager deals with GitLab repositories. In the past, the section had problems with critical pieces of software not checked into version control, or not having their repositories synchronized with the code that was effectively in production. The bipy-gui-manager addresses the issue by 1) setting up the repository for the developers, so even if they don't know or have no time for version control, the tool takes care of it, and 2) by not allowing the developers to use its simplified release function unless they commit all their changes to GitLab. It is important here to note that bipy-gui-manager does not really block the developer from releasing uncommitted code: a slightly more expert person can still do a release in a single (although longer) command. However, we believe that such small hurdles will make programmers follow conventions, which in turn will help lower our code hand-over and maintenance efforts. This is especially true for projects made by newcomers and interns, who are often tasked with developing or maintaining expert GUIs as a way to

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

familiarize themselves with systems they will eventually be working on.

The bipy-gui-manager is already in use in the section for our first Python Expert GUIs.

RAD Tools: ComRAD

Along with the development of Acc-Py, the team produced another interesting tool for Rapid Application Development (RAD), called ComRAD[5]. This paper will not go into detail about the implementation details of this zero-code GUI tool, but rather highlight how it has contributed to our PyQt efforts.

While initially aimed only at prototypes and fixed-display (non interactive) applications, its scope has been gradually expanded, and it has eventually became an interesting tool for general Expert GUI development.

Therefore, we decided to perform an evaluation of this tool and understand which role it could take in our standard PyQt development workflow. Our requirements for a successful evaluation of this tool included:

- Do not limit the developers from using the full capabilities of PyQt. This is important to allow Expert GUIs to grow in complexity if the need arises.
- Allow the use of custom widgets, to be able to extract components and reduce the development time.
- Be able to package and deploy ComRAD applications on our NFS file system as a regular Python application.
- Have some actual benefit over bare PyQt. This was important to avoid adding unnecessary complexity to our tools without any effective gain.

Our analysis was generally positive: ComRAD matched all our requirements and exceeded them by cutting development time for simple Expert GUIs to hours rather than days.

Consequently, ComRAD was added to our workflow. Such an addition was performed by adapting bipy-gui-manager, which is now capable of creating both PyQt and ComRAD oriented Expert GUI projects.

This operation also reinforced the relevance of bipy-gui-manager for controlling our workflow: the tool was easy to adapt to the new standard, and enabled all our developers to develop ComRAD based applications almost immediately at the end of our evaluation and the decision to make it available to developers.

A LANGUAGE-AGNOSTIC LAUNCHER

One pillar of our Expert GUI ecosystem is the AppLauncher[6] (Figure 1), a tool which provides two main functionalities: a) it lets us manage user access to expert GUIs, and b) it provides users a centralized catalogue of the expert GUIs a user has access to.

On the operational computers in CERN's Control Center, the console managers point not to the application's binary itself, but to the AppLauncher executable, with a parameter set to the desired application name. This layer of abstraction allows on one side the operators to have a reliable entry point to find applications, and on the other hand gives the developers more freedom on where and how to install their applications for operational use, so long as they can be registered in the AppLauncher and be launched on the target machine.

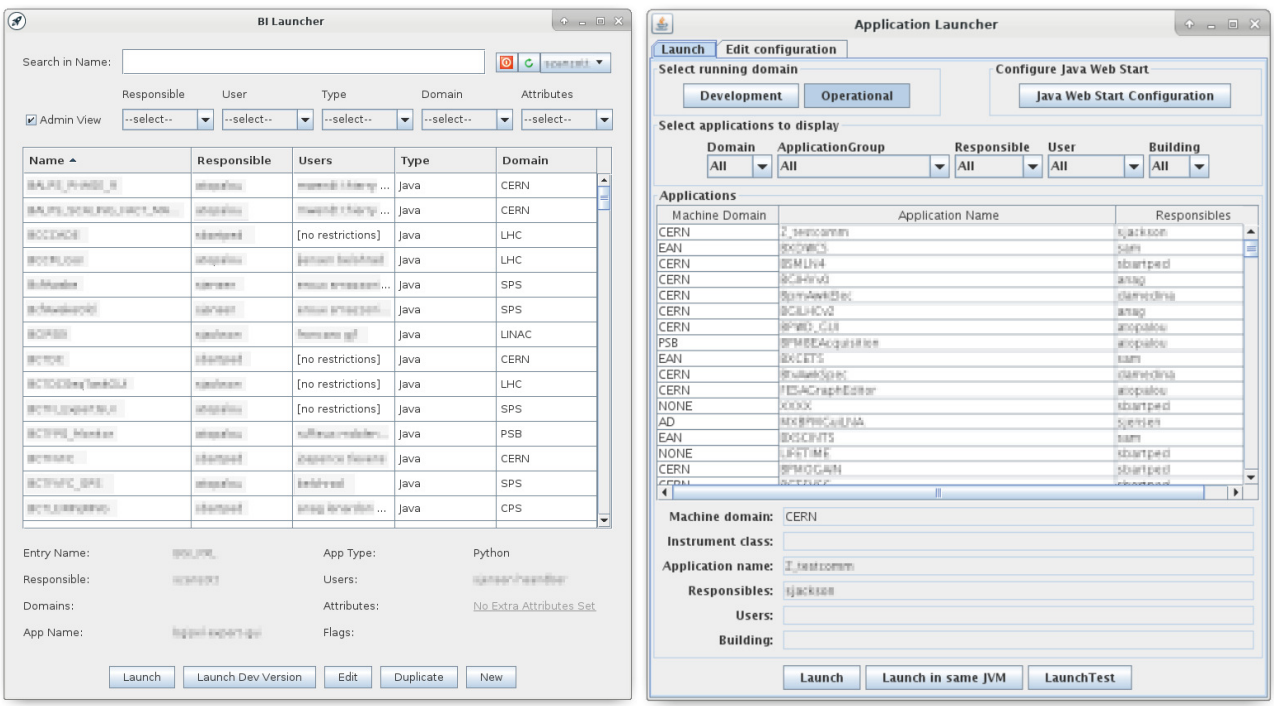


Figure 1: The BI Launcher (left) and the old AppLauncher (right).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV014

Given the nature of expert GUIs, accidental misuse could lead to serious equipment damage; however, the AppLauncher did not enforce any strict access control, and it was rather oriented towards helping users finds the correct GUI faster access rather than limiting access to them. In addition, being designed to support Java applications only, it was relying at its core on the Java Web Start system, which has been preserved at CERN after its demise by Oracle as an in-house tool called JWS. The AppLauncher, although rather simple, could not be repurposed to support any other language, and therefore this option was excluded in favour of a more radical one: a full rewrite. The new AppLauncher, now called simply BI Launcher, is a new Java Swing application that imitates to some degree the look and feel of its predecessor, while fundamentally rethinking its internals. By adopting more modern Java standards and reducing the features to what the section mostly uses this tool for, the BI Launcher ended up as a simpler application with a compact codebase, able to launch Java (though JWS), Python (through Acc-Py tools) and Web pages (through the browser) from the same interface. It also modernised the security aspects of accessing the application catalogue, by utilising CERN's widely used role-based-access system (RBAC[7]).

The old AppLauncher was eventually decommissioned in September.

FIRST PYTHON EXPERT GUIS

PyQt Expert GUIs

In parallel to the work detailed above, we also ported a few Expert GUIs to PyQt to use them as a proof of concept, and to identify potential blocking issues that we might have overlooked. Candidate GUIs for this role had to match the following requirements:

- *Be small:* in order to be able to quickly iterate in case of issues, a good candidate GUIs should be small, both in the expected codebase size and in the interface.
- *Be thin*: we are not interested at this stage in GUIs that contain a lot of logic unrelated to the interface (like complex calculations or domain-specific algorithms). We need GUIs that do nothing more than allowing the user to read, write and monitor values.
- *Already due for a rewrite*: some GUIs were already in need of a full rewrite. For example, we have a number of JavaFX-based GUIs which we want to migrate, due to the fact that JavaFX is no more a recommended technology for GUIs. Other examples include GUIs based on Swing or Qt (C++), developed by other teams but assigned to our section for maintenance.
- *Include commonly used widgets:* expert GUIs tend to include common widgets, like plots, timing displays, log viewers, toggles, etc. Testing the new technology on these components was mandatory.
- *Be actively used*: the ultimate test is operational use. We opted for GUIs being actively used, rather than made-up test cases.

After an inspection of the collection of GUIs that matched the above requirements, we selected a couple of them for the initial port. The chosen applications were:

- A medium-small application already partially ported to PyQt by an intern in a previous evaluation of the technology, which we are going to identify as "BGI App";
- A very small JavaFX application with some interesting features and plot customizations, which we are going to refer to as "BCF App" (Figure 2).

We began the process from the BGI App in an attempt to define a good project template to be followed by other applications. We defined the features at a project level (GitLab repo, test suite, CI, README, code structure, etc) but also at a visual level (mandatory widgets like the log panel at the bottom of the window, timing bar at the top, etc…) and used the resulting template in the bipy-gui-manager to ensure uniformity for future applications.

Once this step was complete, we moved on to the BCF App, the first real port from scratch. In this case we started with the newly created template, and we verified that the bipy-gui-manager was an effective tool for the task. Then we proceeded with the actual port and succeeded in producing a fully functional application which replicated faithfully all the features present in its JavaFX counterpart.

In the process of porting both application we also produced a number of reusable PyQt widgets that, especially once repackaged in BE-CSS widget libraries, are going to make building future PyQt applications even faster and simpler than it was for these ones. Examples of these widgets include the Timing Bar, a widget that displays timing information of a selected accelerator, the Log Console, a highly customizable widget that receives, displays and filters the logs, the crosshair, that was added to the basic plots widget after our example, plus several others that are still being evaluated.

ComRAD Expert GUIs

Shortly after porting the two Expert GUIs mentioned above, ComRAD became ready for evaluation. As a test case, we chose to migrate and simplify a C++ Qt-based application called "PXL App".

Our experiences with ComRAD were considered highly successful, due to a series of benefits that we identified while working on the application:

- *Faster development*. The initial iterations of the GUI were very fast and required barely any code, due to the great amount of Qt UI files that could be reused from the C++ based counterpart. In addition, ComRAD provides useful CERN-specific widgets which facilitated shorter iterations.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV014

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

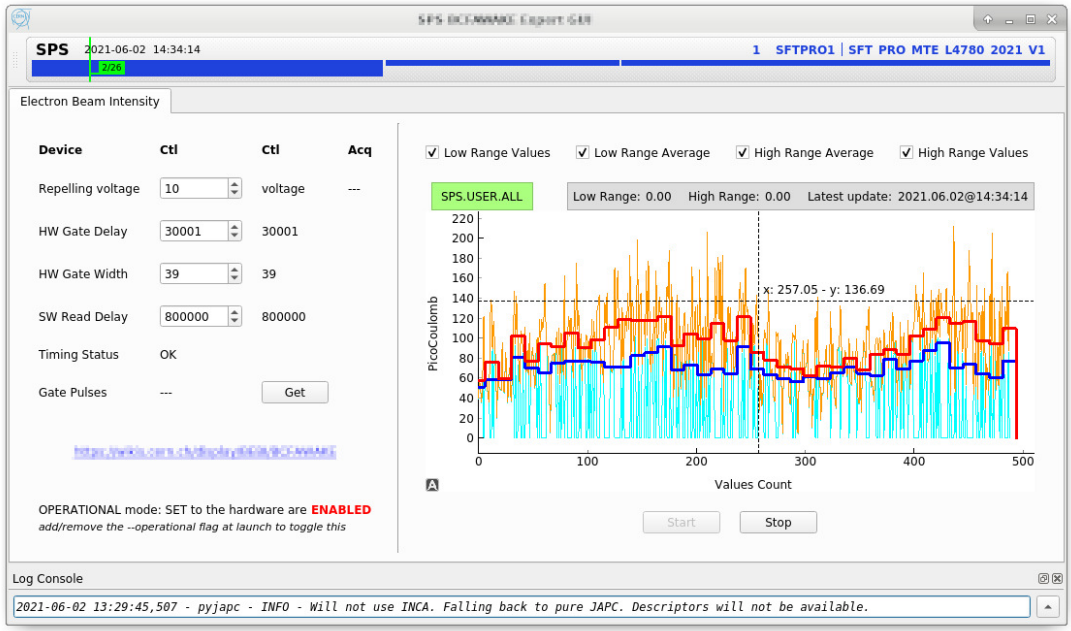


Figure 2: Example of an expert GUI ported to PyQt from JavaFX: the BCF App.

- *Decoupling interface design and interface logic.* ComRAD enabled us to design the interface with no boilerplate code to support it, through enhancements made on the Qt Designer. This decoupling allowed us to first build an application with no connections to the middleware and iterate on the design without the need to think about the code.
- *Progressive wiring.* ComRAD allowed us to progressively connect widgets from the simplest to the most complicated. In addition, for the majority of the widgets, it required no code at all to connect to the control system. This code-less approach could be used also in simple plot widgets, and code was required only for the most complex ones.
- *Allows pure PyQt code*. ComRAD was designed to allow developers to fall back to a pure PyQt-style application where needed, while allowing the rest of the application to benefit of its RAD facilities. In fact, while ComRAD has its own enhanced Designer files, it can read and manage regular Qt Designer interfaces too; while it has its own highly automated system to connect to the middleware, it also allows developers to use their own connections, and so on. This is a very strong feature, as it means that the framework will not hinder the developer if they need to implement something different from what ComRAD was designed for, which means that it will be easier to support in the long-term, and even to replace at a future date if the need arises.

Performance Assessment

One of the most worrying concerns of porting our Expert GUIs to Python-based technologies was the performance of some demanding widgets, for example tables and plots. While we met no performance bottlenecks during the development and operation of the first two PyQt GUIs, we finally faced the problem with the PXL App, when we ported it to ComRAD. This app was initially written in C++, and not in Java, due to several reasons: some demanding plots, hundreds of widgets to display, and a special C++ library being used in the backend. Although simplified in its ComRAD version, this GUI still featured a plot that was required to fully re-render up to 1.048.576 points (1024 lines with 1024 data points each) in less than a second in order not to freeze or skip updates.

We soon realized that such requirement could not be satisfied, and that the GUI would be able to barely deal with 400 such lines (409.600 points). However, after presenting the issue to the users of this application, we realized that there was no need to display the entire bulk of the data and that displaying a subset of 100 lines was sufficient. With this new concession, the application was able to run smoothly.

FUTURE WORK

Now that the technology is proven and some GUIs have been successfully ported, the next steps involve primarily the port of all the remaining GUIs that need a rewrite. This port will not be simply a translation of the old codebase into a new one: at every iteration we plan to assess which components can be transformed into generic widgets, extract them, package them, and release them to be reused as ComRAD widgets for the following applications.

We believe this process will benefit both ourselves, by speeding up the development of more complex application, and the wider PyQt and ComRAD community at CERN.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV014

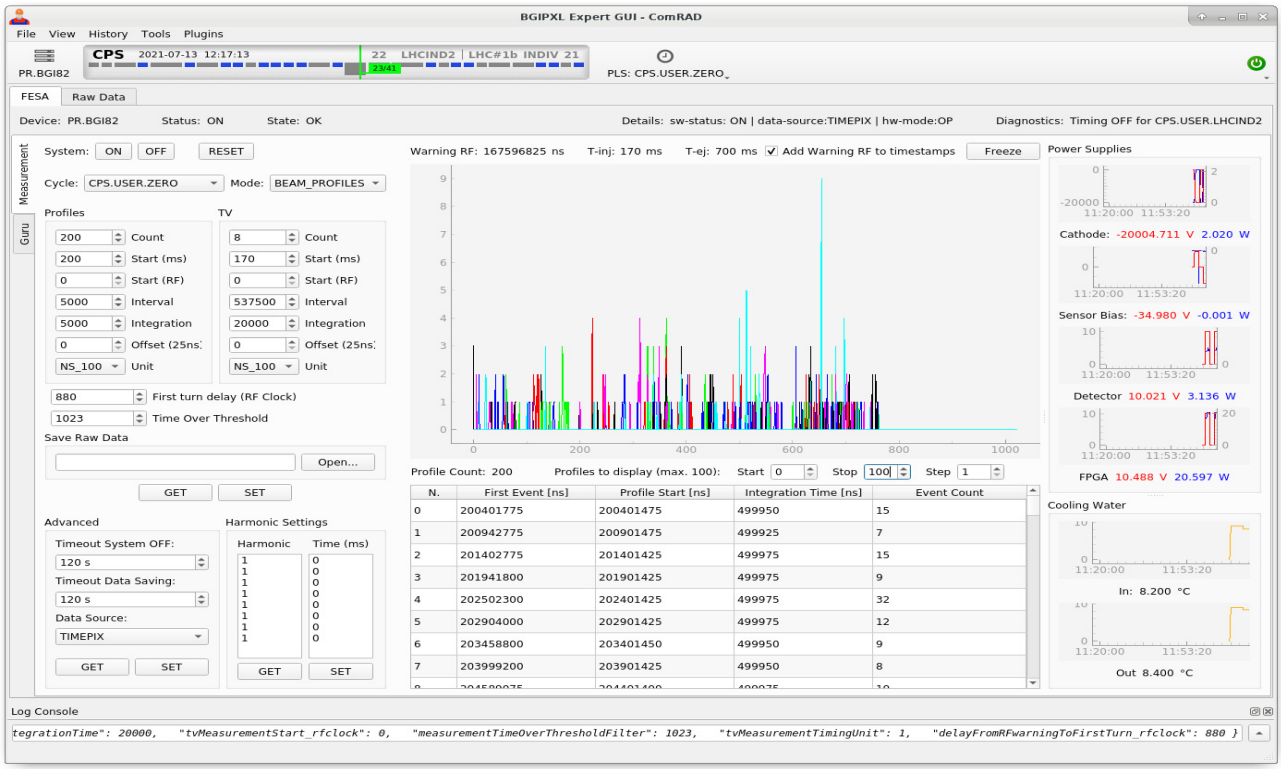


Figure 3: Example of an expert GUI ported to ComRAD from C++ Qt: the PXL App.

CONCLUSION

Adopting PyQt as the main framework for Expert GUIs development in our group was not a straightforward task. Most of the assumptions made in the past about the applications had to be re-evaluated to allow non-Java based GUIs to enter the scene. With the help of another team (BE-CSS) which took care of the middleware integrations, we progressively cleared the path for seamless PyQt GUI development by: a) modifying the AppLauncher to enable launching of any type of GUI application, b) developing the bipy-gui-manager to promote standardization and best practices within our team c) porting two pilot GUIs to PyQt, d) porting one GUI to ComRAD and e) assessing the performance capabilities of the plots in a demanding real use-case.

While the future for PyQt as the main technology for future GUIs is still not fully clear, it is clear that PyQt will be a key technology for RAD purposes. In this context, the work put in place to maintain, control and catalogue these efforts will be valuable regardless of whether we decide to move our entire portfolio of GUIs to PyQt technologies or not, and we expect this workflow to serve us well in the years to come.

REFERENCES

[1] S. Bart Pedersen and S. Jackson, "Graphical User Interface programming challenges moving beyond Java Swing and JavaFX", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 637–640. doi:10.18429/JACoW-ICALEPCS2019-MOPHA173

[2] I. Sinkarenko, S. Zanzottera, and V. Baggiolini, "Our journey from Java to PyQt and Web for CERN accelerator control GUIs", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 807-811. doi:10.18429/JACoW-ICALEPCS2019-TUCPR03

[3] S. Zanzottera, "Evaluation of Qt as GUI framework for accelerator controls", M.Sc. thesis, Politecnico di Milano, Italy, 2018.

[4] S. Zanzottera, "Status of Python for GUIs", https://indico.cern.ch/event/1031183/contributions/4330130/attachments/2239943/3797572/Status%20of%20Python%20for%20GUIs.pdf

[5] I. Sinkarenko, "Python GUI Status Update" https://indico.cern.ch/event/1025056/contributions/4303998/attachments/2235763/3789373/Python%20GUI%20Status%20Update.pdf

[6] P. Karlsson and S. Jackson, "The Introduction of hierarchical structure and application security to Java Web Start development", in *Proc. ICALEPCS'05*, Geneva, Oct 2005, paper TH3A.2-5O

[7] K. Kostro, W. Gajewski, and S. Gysin, "Rolebased authorization in equipment access at CERN", in *Proc ICALEPCS'07*, Knoxville, Tennessee, USA, Oct. 2007, paper WPPB08, pp. 415-417.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV015

NEW TIMING SEQUENCER APPLICATION IN PYTHON WITH Qt DEVELOPMENT WORKFLOW AND LESSONS LEARNT

Z. Kovari, G. Kruk, CERN, Geneva, Switzerland

Abstract

PyQt is a Python binding for the popular Qt framework for the development of desktop applications. By using PyQt one can leverage Qt's aspects to implement modern, intuitive, and cross-platform applications while benefiting from Python's flexibility. Recently, we successfully used PyQt 5 to renovate the Graphical User Interface (GUI) used to control the CERN accelerator timing system. The GUI application interfaces with a Java-based service behind the scenes. In this paper we introduce the generic architecture used for this project, our development workflow as well as the challenges and lessons we learnt from using Python with Qt. We present our approach to delivering an operational application with a particular focus on testing, quality assurance, and continuous integration.

TIMING CONTROL APPLICATION

Accelerator Timing System

CERN continuously delivers particle beams to a range of physics experiments (end-users), each posing strict, detailed requirements with respect to the physical characteristics of the particle beam to be delivered. Hence, particle beams traverse a number of accelerators while being manipulated in various ways. For this to happen, the accelerators repeatedly "play" pre-defined cycles, which usually consist of an injection-acceleration-ejection sequence. This in turn involves many concurrent beam manipulations including particle production, bunching, cooling, steering, acceleration and beam transfer – all of which must occur at precise moments in time, often with microsecond or even nanosecond precision. The role of the Timing system is an orchestration of all these activities, ensuring that the accelerator complex behaves as expected as a function of time.

Each accelerator at CERN is associated with a Central Timing system (CT) which, based on the configuration provided by the operators and dynamic input such as external conditions, calculates in real time so-called *General Machine Timing (GMT)* events that define key moments in the accelerator cycles such as beginning of the cycle, injection, ramp, extraction, etc.

GMT events are then transmitted to *Front-End Computers (FECs)* via a dedicated cabled network known as the *GMT network*.

On the FEC side, the GMT cables are connected to *Central Timing Receiver* (CTR) modules, which decode the received GMT events and allow generation of derived local events (with optional delay) in the form of software interrupts and physical pulses for the accelerator hardware.

Timing App Suite

The control of the timing system is done via a dedicated GUI application that allows operation crews to define a collection of cycles composing a so-called *beam* (see Fig. 1). A Beam is executed by the central timing to transfer and accelerate a particular particle beam from the source, through the intermediate accelerators, up to the final experiment. Beams are then used to build a *Beam Coordination Diagram (BCD)* that defines sequencing of different particle beams sent to different destinations as illustrated in Fig. 2.

Following the renovation of the central timing itself, it was decided to also renovate the 20-year-old application used to control it, modernizing its architecture, improving usability aspects and taking advantage of the new features provided by the central timing.

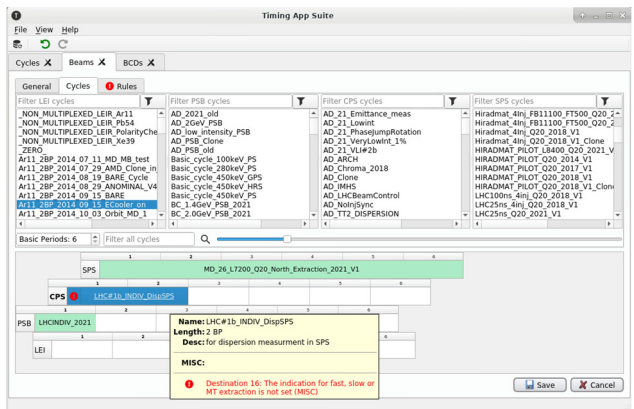


Figure 1: Timing *Beam* editor.

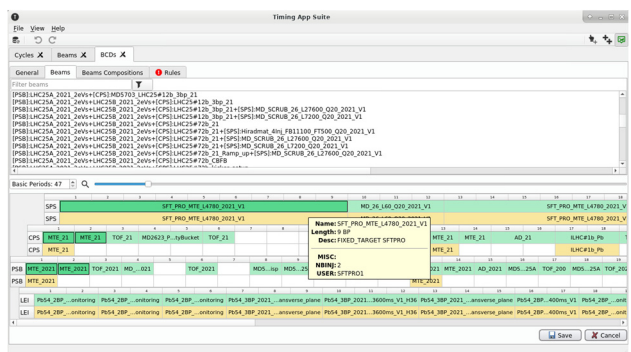


Figure 2: *Beam Coordination Diagram* editor.

ARCHITECTURE

The application was designed and implemented in a 3-tier architecture (see Fig. 3): a GUI, implemented in Python using PyQt toolkit, communicating via HTTP with RESTful services implemented in Java, and with the Cen-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV015

tral Timing process (C++) via PyJAPC [1] a Python wrapper over the Controls Middleware (CMW) [2] communication library.

The Java server relies on Spring Boot and several features provided by the Spring framework such as REST controllers or JPA database access, along with dependency injection and configuration services.

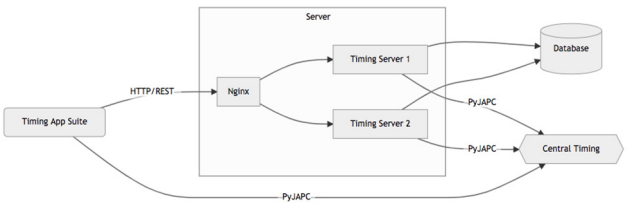


Figure 3: Timing App Suite architecture.

The server is completely stateless allowing simple deployment of two instances with an Nginx [3] proxy server in front, and enabling rolling updates.

GRAPHICAL USER INTERFACE

The GUI application is split into four different layers:

- ***Data***: Qt independent CERN-related domain classes representing the Timing ecosystem.
- ***Model***: Qt model layer responsible for mapping Data to the View [4].
- ***View***: the User Interface widgets [4].
- ***Controller***: optional intermediate layer between view and model, typically handling signal and slot connections that need more sophisticated business logic and cannot be connected directly between the view and the model.

Model

The model layer in Qt's architecture has two responsibilities. On the one hand, it maps the Data objects to the View, basically saying which data item should be displayed at which index (*QModelIndex* [5]). On the other hand, it also tells the View component how that data should be rendered, e.g., in which font, background color, and with what tooltip or icon.

View

In most cases, the user interface was designed in Qt's built-in editor, *Qt Designer,* which is also easily accessible through the pyqt5-tools [6] Python package. *Qt Designer* is a simple-to-use editor that allows dragging and dropping Qt widgets and creating user interfaces without any programming.

Such created panels are then saved to *.ui* files, which are re-used in the Timing Control Python application. There are two ways to do that:

- Using PyQt's *uic* module to load the generated UI files [7].
- Generating Python code from the UI files [8].

The 2nd solution was used, as this approach provides content assists in the *PyCharm* IDE, and in general, helped to better integrate the View components into the source code (e.g. finding usages throughout the entire source code).

The drawback of such an approach was that an additional step was needed between editing the UI in Qt Designer and then using it in the application. The *pyqt5ac* 3rd party library was used for code generation to automatically convert multiple UI files into Python [9].

Communication Between Model and View

The model layer in Qt handles many aspects of what is displayed, and how, from *Data* to *View*. Due to this close connection, it is also simple to directly handle user interactions between view and model by connecting the necessary signals from the view to the model's slots, and vice-versa. That being said, an additional controller layer was introduced to handle some of the more complex signal-slot connections. This also allowed to further modularize the application and re-use some of the view, model, or even controller components in other panels.

Testing

One of the great benefits of using Qt/PyQt was automated testing. Using the *pytest-qt* Python library [10], over 500 test cases were implemented that verify the user interface. The *pytest-qt* package provides a *qtbot* object that can be used to spawn and interact with the GUI application, e.g., clicking buttons, selecting, or even dragging and dropping elements. Behind the scenes, the package uses Qt's *QTest* namespace [11] with some additional features extended (e.g., "stop" method [12]).

TIMING JAVA SERVER

The server side follows the classical pattern with three layers [Fig4]: (1) the data access layer consisting of JPA repositories and corresponding entities mapped to the database tables, (2) the service layer that uses one or more repositories to perform CRUD operations and does the translation between entities and Data Transfer Objects (DTOs), and (3) the REST controllers layer that handles HTTP requests, relying on services.

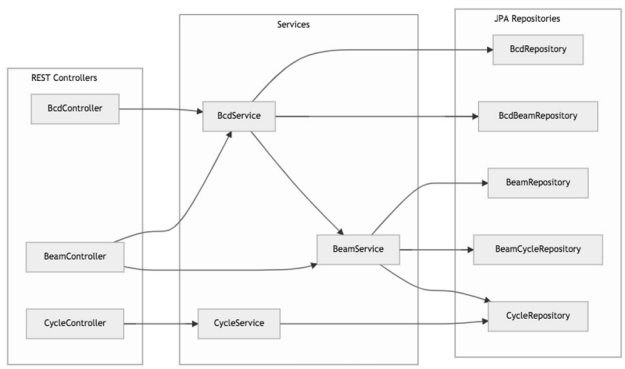


Figure 4: Simplified diagram of Timing server components.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

CONTINUOUS INTEGRATION

CERN's self-hosted *GitLab* instance was used to host the application's code repository and set up automated pipelines [13] for continuous quality assurance.

Each time a commit is pushed to the Git repository, a *GitLab* pipeline (see Fig. 5) runs and automatically verifies the codebase of that specific commit. Verification means the following:

- GUI tests run automatically on *GitLab*. To run UI tests in a container, Xvfb [14] (a virtual framebuffer X server that can run on machines with no display) was configured.
- Static code analyzer tools (such *flake8* [15], *pylint* [16], and *mypy* [17]) are used to analyze the correctness of the code. These tools can detect non-standard code usages that violate PEP-8, or even potential problems such as missing arguments or missing imports. Based on the development experience, these tools are easy to set up and do help to elevate code quality but don't replace frequent code reviews and testing.

Figure 5: Pipeline to build, test and analyse the code.

CONTINUOUS DEPLOYMENT

On the same *GitLab* pipelines that verify the code, the application can also be deployed to different environments, as illustrated in Fig. 6.

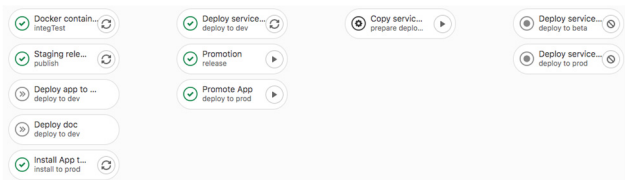


Figure 6: Deployment part of the pipeline.

To track the environments (such as DEV, STAGING, RC, PROD) the *GitLab* built-in environments feature is used [18]. *GitLab's* environments panel allows to track which commits are deployed to which environments. Furthermore, a deployment pipeline can also be quickly triggered from the same panel, and likewise a roll back to a previous version if necessary.

Application deployment is carried out with *Ansible* [19]. The Python application's wheels [20] are built on *GitLab CI*, dependencies are locked, and then with *Ansible*, binaries are transferred to the deployment server where the Python virtual environment is re-created and the application deployed.

The deployment of the backend server is also triggered from the same pipeline. The server and application code are stored in the same repository, which allows to version and deploy them together to production.

Docker images are built which contain the backend server's binaries and then deployed with *Ansible* to the various environments (DEV, STAGING, RC, PROD). One such environment comprises a *Podman* pod [21] with three running containers: two for the server to facilitate rolling updates, and one *Nginx* container to delegate the incoming requests between the two other containers.

CHALLENGES AND LESSONS LEARNT

Python for GUI

From the end of 90's until 2020, the vast majority of CERN Controls desktop applications were implemented in Java, for many years using the Swing toolkit, and more recently JavaFX.

For the new Timing application, it was decided to switch to Python and PyQt. The decision was not easy and was mainly dictated by the announcements made by Oracle for deprecation of Swing and discontinued support for JavaFX. Both frameworks are unlikely to disappear within the next few years (Swing will still remain a part of the JDK at least over the next few years), however their future beyond the time frame of 8-10 years is very uncertain. The lifetime of CERN controls applications (including Timing App Suite) is typically beyond 20 years, therefore for the development of new GUI software, requiring significant investment, it was preferred to switch to a framework that is actively developed and maintained, and whose future seems to be much more reassuring.

Python is a powerful, flexible, easy-to-learn and easy-to-use language. One needs less lines of code to perform the same task when compared to other major languages like C++ or Java, which turns into a better productivity. These advantages apply also to GUI programming. The biggest issue confronted when developing the application comes from the flexibility of the language i.e. dynamic typing and lack of compilation checks.

While the codebase of the application was growing, many internal modifications were applied, such as renaming classes, functions and variables, moving parts of the logic from one place to another, replacing one class with another etc. *PyCharm* is dealing with most of the refactoring quite well, but occasionally was failing to properly identify the type of a variable and not applying the requested change. After realizing this, type hints were consistently used all over the code, paying particular attention to function declarations i.e. parameters and return types.

The static analysis tools came in very useful to catch various issues related to the code refactoring, however the ultimate safety net to protect from these kinds of risks was the high coverage of the code by automated tests.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV015

PyQt Toolkit

Qt provides all standard components designed following the model/view architecture, therefore their usage did not pose any major problems. The concept of signals and slots for establishing communication among various components permits flexibility when dealing with GUI events and results in a smooth codebase.

PyQt is one of the most used UI frameworks for Python therefore there is a relatively big community behind it. There are many learning resources for Qt and PyQt, however a complete and up-to-date PyQt beginners tutorial, with simple and more advanced code examples was found to be missing. Answers to most specific questions could be found, but this required browsing through different forums and blogs, and often applying solutions given for C++ Qt or PyQt4, to the PyQt5-based application.

There is a certain learning curve, especially when it comes to a bit more advanced usage of UI widgets, although once foundations were established and the concepts became familiar, the Qt API documentation was deemed satisfactory most of the time.

The documentation of the testing library, pytest-qt, is rather modest, so learning it and setting up bases for UI tests required a bit of time to experiment and browse Qt forums.

On several occasions segmentation faults (coming from the Qt widgets) were encountered. At first these were not straightforward to understand, but with time they could be interpreted, and it quickly became possible to identify their source.

Architecture

Implementing the GUI in Python and the backend in Java did not pose any issues. The advantages of both languages and corresponding frameworks could be leveraged to develop the adequate logic in the most efficient way.

What comes as a drawback in this kind of approach is the duplication of domain objects (or Data Transfer Objects) as they are needed in both languages. The Timing API is relatively small and the domain objects quite simple, therefore implementing and maintaining them in Java and Python was a negligible effort.

CONCLUSIONS

After two decades of developing desktop controls applications using Java, it was decided to change the programming language to Python with PyQt selected as the widget toolkit. The choice was not obvious, but the end result is more than satisfactory, and the same approach is now foreseen for some other applications e.g. the CERN developed Front End Software Architecture [22] diagnostic tool (known as the FESA Navigator).

Efficiency and flexibility, while preserving readability of Python-based applications as compared to Java, comes along with the lack of compile time verification of the codebase. Therefore, a rigorous approach to Continuous Integration performing static code analysis, together with high coverage by automated testing is deemed indispensable for any operational application.

REFERENCES

[1] V. Baggiolini *et al.*, "JAPC - the Java API for Parameter Control", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper TH1.5-8O.

[2] J. Lauener and W. Sliwinski, "How to Design & Implement a Modern Communication Middleware Based on ZeroMQ", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 45-51. doi:10.18429/JACoW-ICALEPCS2017-MOBPL05

[3] Nginx, https://www.nginx.com

[4] Qt5, https://doc.qt.io/qt-5/model-view-programming.html

[5] Qt5, https://doc.qt.io/qt-5/qmodelindex.html

[6] PyQt5 Tools, https://pypi.org/project/pyqt5-tools/

[7] Importing the UI File in Python, https://nitratine.net/blog/post/how-to-import-a-pyqt5-ui-file-in-a-python-gui/#importing-the-ui-file-in-python

[8] First Steps with Qt Designer, https://www.pythonguis.com/tutorials/first-steps-qt-creator

[9] PyQt5 Auto Compiler, https://pypi.org/project/pyqt5ac

[10] PyTest Qt, https://pypi.org/project/pytest-qt

[11] Qt5 Test, https://doc.qt.io/qt-5/qtest.html

[12] QtBot, https://pytest-qt.readthedocs.io/en/latest/reference.html#pytestqt.qtbot.QtBot.stop

[13] CI Pipelines, https://docs.gitlab.com/ee/ci/pipelines

[14] PyTest GitHub Actions, https://pytest-qt.readthedocs.io/en/latest/troubleshooting.html#github-actions

[15] Flake, https://pypi.org/project/flake8

[16] Pylint, https://pylint.org

[17] Mypy, https://mypy.readthedocs.io/en/stable

[18] CI Environments, https://docs.gitlab.com/ee/ci/environments

[19] Ansible, https://www.ansible.com

[20] Python Wheels, https://realpython.com/python-wheels

[21] Podman, https://mohitgoyal.co/2021/04/23/spinning-up-and-managing-pods-with-multiple-containers-with-podman

[22] M.Arruat *et al.*, "Front-End Software Architecture", in *Proc.* in *Proc. ICALEPCS'07*, Knoxville, Tennessee, USA, Oct. 2007, paper WOPA04.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV021

TATU: A FLEXIBLE FPGA-BASED TRIGGER AND TIMER UNIT CREATED ON CompactRIO FOR THE FIRST SIRIUS BEAMLINES

J. R. Piton, D. Alnajjar, D. H. C. Araujo, J. L. Brito Neto,
L. P. do Carmo, L. C. Guedes, M. A. L. de Moraes,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

In the modern synchrotron light sources, the higher brilliance leads to shorter acquisition times at the experimental stations. For most beamlines of the fourth-generation source SIRIUS (the fourth-generation particle accelerator in Campinas, Brazil), it was imperative to shift from the usual software-based synchronization of operations to the much faster triggering by hardware of some key equipment involved in the experiments. As a basis of their control system for devices, the SIRIUS beamlines have standard CompactRIO controllers and I/O modules along the hutches. Equipped with an FPGA (Field Programmable Gate Array) and a hard processor running Linux Real-Time, this platform could deal with the triggers from and to other devices, in the order of ms and µs. TATU (Time and Trigger Unit) is an FPGA/real-time software combination running in a CompactRIO unit to coordinate multiple triggering conditions and actions. TATU can be either the master pulse generator or the follower of other signals. Complex trigger pattern generation is set from a user-friendly standardized interface. EPICS (Experimental Physics and Industrial Control Systems) process variables [1], by means of LNLS Nheengatu [2], are used to set parameters and to follow the execution status. The concept and first field test results in at least four SIRIUS beamlines are presented.

INTRODUCTION

LNLS Tatu (Timing and Trigger Unit), being mostly FPGA code running in a CompactRIO (CRIO) device (NI CRIO models 9045/9049/9035/9039), combines C-Series I/O modules to work as trigger detectors, pulse generators and a recorder of analog and digital input readouts. Tatu manages digital signals to detect events and to produce actions for synchronized operations at a beamline. That comprises sequences of operations on shutters, motorized devices and detectors, allowing the data acquisition at beamlines to be as fast as possible (also known as "fly-scan"). An example of signal combination to produce different outputs is shown in Figure 1.

Some premises that have been considered in the pursuing of this project are as follows:

- it was already decided that each hutch in a SIRIUS beamline would count on at least one device to make available TTL signal input/output and CompactRIO was the selected standard.
- additional small pieces of software, either in FPGA or Real-Time, could be included for dedicated jobs (like filtering analog readouts), in parallel to the triggering management.
- the access to the configuration parameters would happen through Nheengatu [2], so their manipulation would be immediately made available under EPICS.
- at least for the initial operation of the first beamlines, a maximum pulse rate of 4 kHz would fully meet the requirements.
- it should be highly flexible to let different conditions and outputs to be combined, just by software configuration (no FPGA recompilation needed).

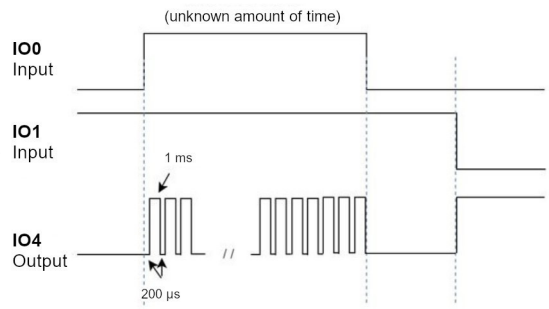


Figure 1: Two cases of output in IO4 port (pulses or "on") depending on the combination of two input ports IO0 and IO1.

THE HARDWARE

Tatu is embedded in an 8-slot CompactRIO controller, under a Real-Time Linux operating system and a programmable FPGA (through LabVIEW). Tatu has been implemented in four different models:

9045 (or 9035) – 1.30 GHz Dual-Core CPU, 2 GB DRAM, Kintex-7 70T FPGA

and the larger

9049 (or 9039) – 1.60 GHz Quad-Core CPU, 4 GB DRAM, Kintex-7 325T FPGA.

The chosen CompactRIO chassis can host C-series modules, a suite of interfaces that provide several analog or digital I/O channels, in a range of different sample rates.

One of two modules can be used under Tatu:

9401 – a 5 V/TTL digital module with 8 bidirectional I/O channels. Four channels are set to input (IO0 to IO3) and four channels to output (IO4 to IO7). In this configur-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV021

ation, the maximum signal switching frequency for the module is 16 MHz per input channel and 10 MHz per output channel [3].

or the larger (but 70 times slower) module

9403 – a 5 V/TTL digital module with 32 bidirectional I/O channels. The first block of four channels is set to input and the next four channels are set to output and so on (IO0 to IO3, IO8 to IO11 and IO16 to IO19 as inputs, IO4 to IO7, IO12 to IO15 and IO20 to IO23 as outputs). Only 12 of 16 input ports and 12 of 16 output ports have been implemented so far, to save FPGA resources. The signal switching frequency is about 150 kHz. [4]

The nominal maximum switching frequency as described in the specification of 9401 and 9403 modules is not reached by Tatu, as its flexibility in the signal management consumes FPGA some clock cycles. Also, our selection of the CRIO chassi model depends on the I/O modules to be used, as they require different demands for FPGA resources. Resource usage and achieved performance in each case is listed in Table 1.

Table 1: CompactRIO FPGA Use and Module Rates

CRIO model	I/O model	FPGA use (Total slices)	I/O rate
9049	9401	24.7 %	10 MHz
	9403	49.2 %	140 kHz
9045	9401	86.8 %	10 MHz

SIGNAL MANAGEMENT

Tatu as the Master Signal Generator

There are some cases where the synchronization of the devices can be provided by Tatu as the signal manager, identifying the status of each involved device and providing trigger signals at the right time, to coordinate their operations as previously set by the user. In this situation, Tatu is said to be in the *Master* mode. Tatu produces a series of internal "master" trigger signals (defined by a given number of pulses, with a duration and periodicity, all previously set by the user). That determines the pace of the operation, being followed by Tatu itself and/or replicated to other devices involved in the data acquisition.

Tatu as a Signal Follower

There are cases where some devices mandatorily require the use of its proprietary software. In such scenarios, the device is the "master" giving the pace of the acquisition operation and Tatu is just another device in the chain. Tatu is set to wait for some external signal and occasionally to replicate it to third parties, other devices in the acquisition. This is the case of Tatu being in the *Follower* mode.

TATU OPERATIONS

When Tatu is deployed, an EPICS IOC (Input Output Controller) is configured and it runs under the Linux RT CompactRIO operating system. It provides several EPICS PVs (process variables) to set and monitor the Tatu operations.

For each input port (IO0...IO3) a trigger state can be detected. The Input Trigger detection modes currently available are:

- an *Up* or *Down* state
- a *Falling edge* or a *Rising edge*
- *under* or *over an Analog threshold (*of a channel in an associated C-series analog module)
- just following the internal *Tatu Master Pulse*
- otherwise, the input is ignored

If one of the *edge* trigger modes is selected, a secondary parameter can be used – *EdgestoTrig.* Then, the triggering condition for input port *n* is satisfied only after the detection of a given number of the specified edge (falling or rising) at that port.

When the input readout meets the trigger mode defined for an input port *n*, the corresponding P*n* state goes true.

Three different cases *c0, c1* and *c2* can be defined for operations on each output port (IO4...IO7). This way, an output port can be turned on in a specific situation and turned off only when another specific condition is present. Each case for an output port has its own *Condition* parameter, whose values are:

- a *Forced* action, as the defined operation should happen regardless of any condition
- the operation should happen as soon as the *Master Pulse* is present
- the operation should happen if any of the P states is true (*P0 ~ P19*)
- the condition is given by a logical combination of *Ps*, so the additional parameter *ConditionCombo* should be inspected to determine the condition; the *ConditionCombo* parameters can be set to one of the logical operations, limited to P0 to P3:
  - *P0 and P1 / P0 and P2 / P0 and P3*
  - *P1 and P2 / P1 and P3*
  - *P2 and P3*
  - *P0 and P1 and P2*
  - *P0 and P1 and P2 and P3*
  - the same combinations with an *or* operator
- otherwise, no output happens

The output will happen after a specific amount of time that can be set (in microseconds) by the *Delay* parameters,

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV021

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

for each case and each output port. The type of output or action (also for each case and each output port) is defined by the *Output* parameters, which can be set to one of the following choices:

- set this output port to 0 *(Off)*
- set this output port to 1 *(On)*
- produce a pulse by setting the port to 1, then to 0 for a given time and to 1 again *(Falling pulse)*
- produce a pulse by setting the port to 0, then to 1 for a given time and to 0 again *(Rising pulse)*
- *Copy* a given value to the output port
- *Copy the inversion* of a given value to the output port
- *Trigger* the scalers readouts and keep them

Whenever the *copy* or the *copy the inversion* is selected, yet another parameter is taken in consideration to define the output operation: which input condition state is to be copied. The *OutputCopy* parameter can bear one of the following values:

- copy the *master pulse* value to the output port
- copy the *P0* state or any of the other 19 states to the output port (P0 ~ P19
- otherwise, no output should be produced

Finally, whenever the output port should produce a falling pulse (or a rising pulse), the duration in microseconds of its low (respectively high) state is given by the *Pulse* parameter.

Tatu operates and produces output only when it is put in the *active* state, through an EPICS PV. From then on, any changes in the PVs for Tatu settings are ignored and the input raw data PVs are not updated anymore. All efforts are put in the Tatu execution, and the follow-up must happen through Tatu status Pvs.

The *master pulse* is an internally produced reference. A user example of Tatu parameters set (in CompactRIO1 of hutch B, Manacá beamline) is shown in Figure 2. That would produce a train of four 250-microsecond pulses in the output port IO7 every half second, 10 times.

```
$ caput MNC:B:RIO01:9401H:Activate  Off (or: 0)

$ caput MNC:B:RIO01:9401H:MasterMode  On(or: 1)
$ caput MNC:B:RIO01:9401H:Zeropulses  On (or: 1)
$ caput MNC:B:RIO01:9401H:MasterPulseNumber  10
$ caput MNC:B:RIO01:9401H:MasterPulseLength  1000
$ caput MNC:B:RIO01:9401H:MasterPulsePeriod  500000

$ caput MNC:B:RIO01:9401H:InputTriggerIO0  Follower (or: 7)
$ caput MNC:B:RIO01:9401H:ConditionIO7:c0  Master pulse (or: 2)
$ caput MNC:B:RIO01:9401H:OutputIO7:c0  Rising pulse (or: 4)
$ caput MNC:B:RIO01:9401H:PulseIO7:c0  200
$ caput MNC:B:RIO01:9401H:DelayIO7:c0  50

$ caput MNC:B:RIO01:9401H:Activate  On (or: 1)
```

Figure 2: EPICS PVs used to configure Tatu in the *master mode* (time units in microseconds).

As soon as Tatu is in the *activate* state, that produces pulses in the output port IO7 (Figure 3).

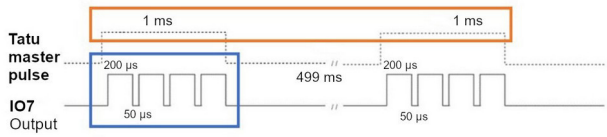


Figure 3: The port IO7 producing output pulses at the *master pulse* pace.

There are watchdog PVs to show that RT-side and FPGA-side software is running. As Tatu gains new features after each new beamline, it may be helpful for the users to check the version control identifier embedded in the FPGA. That can be inspected through an *ai* EPICS PV, a longint number consisting of eight digits *yyyymmdd* (representing *year, month, day)*; for instance, the number **20210131** represents "**January 31st, 2021**".

A small piece of software can be connected to the CRIO to provide a debugging tool, with live values shown (Figure 4).

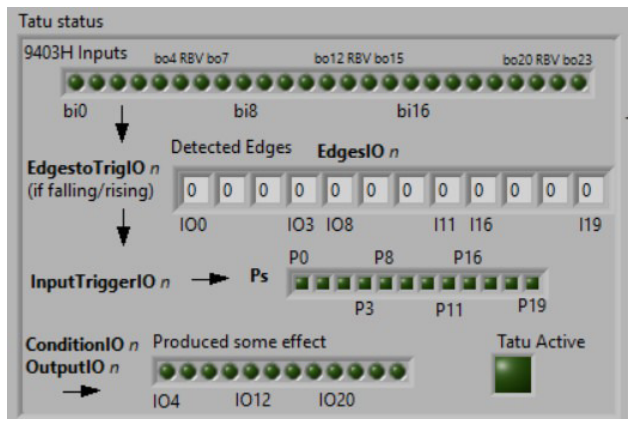


Figure 4: Debug tool for Tatu status.

APPLICATION AT THE BEAMLINES

All beamlines now in commissioning at SIRIUS have one or more Tatu units.

In the commissioning of *Tarumã*, an experimental station at *Carnaúba* (Coherent X-ray Nanoprobe) beamline [5], Tatu has been used for instance in the scanning of areas as small as 80 µm x 80 µm and 800 nm steps. Tatu gives 500 µs pulses along 10100 points, in a total time of 30 seconds for the whole acquisition. [6]

Manacá (Macromolecular Micro and Nanocrystallography) beamline has Tatu in the follower mode, coordinating the shutter opening (1 ms pulse) and the trigger to start the detector in the beginning of an acquisition. This beamline was the first open to user projects [7].

As of Oct. 2021 *Ema* (Extreme condition Methods of Analysis) beamline will start to use Tatu during the energy scanning operations of its high dynamic double-crystal monochromator [8].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV021

CONCLUSION

The premises for such a solution at SIRIUS beamlines were satisfied and Tatu was available in time for the initial tests at SIRIUS *Mogno* beamline (X-ray Micro- and Nanotomography). In sequence, it has been applied for fast data acquisition during the commissioning of the next released SIRIUS beamlines – *Manacá*, *Cateretê*, *Carnaúba*. The flexibility to set parameters and operation modes in Tatu allowed the technical staff to quickly include new input signals and distribute triggering signals to additional devices on demand. An improvement is to admit one or more additional digital I/O modules to increase the number of available ports by keeping the performance observed with just one 100 kHz module. The solution proved successful given the time requirements. Where a better time performance will be needed, the Tatu can be purportedly redesigned, giving up flexibility for the sake of speed. Furthermore, in the future some beamlines will certainly face time requirements too strict to be met by the CompactRIO hardware platform and novel solutions will be selected.

ACKNOWLEDGEMENTS

The authors would like to thank their colleagues among the Brazilian Synchrotron Light Laboratory (LNLS) staff, specially at the groups: SOL-Beamline Software, COI-Beamline Controls and Integration, IEA-Beamline Automation, GIE-Electronics Instrumentation and PLL-Beamline Design for the technical support and advice. During the tests, the contributions of Fernando Henrique Cardoso (GIE), Cassiano Correa Bueno (PLL), Gabriel Schubert (Mogno), Lucca Campoi (Mogno), Andrey Nascimento (Manacá), Tiago Araújo Kalile (Cateretê), João Paulo Zerba (Cateretê), Marcio Paduan Donadio and Gabriel Fedel were much appreciated.

REFERENCES

[1] EPICS-The Experimental Physics and Industrial Control System, http://www.epics-controls.org

[2] D. Alnajjar, G. S. Fedel and J. R. Piton, "Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT," in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, N. York, USA, Oct. 2019, pp. 997-1000. doi:10.18429/JACoW-ICALEPCS2019-WEMPL002

[3] *NI 9401 Datasheet (p/n 374068A-02),* National Instruments, Austin, TX, USA, Dec. 2015, pp. 4-5. http://www.ni.com/pdf/manuals/374068a_02.pdf

[4] *NI 9403 Datasheet (p/n 374069A-02),* National Instruments, Austin, TX, USA, Dec. 2015, pp. 4-5. http://www.ni.com/pdf/manuals/374069a_02.pdf

[5] Hélio C. N. Tolentino, Renan R. Geraldes, Gabriel B. Z. L. Moreno, Artur C. Pinto, Cassiano S. N. C. Bueno, Leonardo M. Kofukuda, Anna P. S. Sotero, Antonio C. P. Neto, Francesco R. Lena, Willian H. Wilendorf, Giovanni L. Baraldi, Sergio A. L. Luiz, Carlos S. B. Dias, Carlos A. Pérez, Itamar T. Neckel, Douglas Galante, Veronica C Teixeira, and Dean Hesterberg, "X-ray microscopy developments at Sirius-LNLS: first commissioning experiments at the Carnauba beamline", Proc. SPIE 11839, X-Ray Nanoimaging: Instruments and Methods V, 1183904 (8 September 2021). doi:10.1117/12.2596496

[6] C. S. N. C. Bueno, G. N. Kontogiorgos, L. Martins dos Santos, L. C. Guedes, L. G. Capovilla, J. R. Piton, A. C. Piccino Neto, R. R. Geraldes, G. B. Z. L. Moreno, M. A. L. Moraes, "Position Scanning Solutions at the TARUMÃ Station at the CARNAÚBA Beamline at Sirius/LNLS", presented at the *18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, Beijing, China, Oct 2021, paper WEPV002, this conference.

[7] G.D. Noske, A.M. Nakamura, V.O. Gawriljuk, R.S. Fernandes, G.M.A. Lima, H.V.D. Rosa, H.D. Pereira, A.C.M. Zeri, A.F.Z. Nascimento, M.C.L.C. Freire, D. Fearon, A. Douangamath, F. von Delft, G. Oliva, A.S. Godoy, "A Crystallographic Snapshot of SARS-CoV-2 Main Protease Maturation Process", Journal of Molecular Biology, Volume 433, Issue 18, 2021, 167118, ISSN 0022-2836. doi:10.1016/j.jmb.2021.167118

[8] R. R. Geraldes, M. A. L. Moraes, R. M. Caliari, E. P. Coelho, L. P. do Carmo, A. Y. Horita and S. A. L. Luiz, "The FPGA-based Control Architecture, EPICS Interface and Advanced Operational Modes of the High-Dynamic Double-Crystal Monochromator for Sirius/LNLS", presented at the *18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)*, Beijing, China, Oct 2021, paper TUPV004, this conference.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV022

MRF TIMING SYSTEM DESIGN AT SARAF

A. Gaget†, CEA Saclay IRFU, Gif sur Yvette, France

Abstract

CEA Saclay Irfu is in charge of an important part of the control system of the SARAF LINAC accelerator based in Tel-Aviv. It's including among other the control of the timing system (synchronization and timestamping). CEA has already installed and used successfully timing distribution with MRF on test bench for ESS or IPHI, so it has been decided to use the same technologies. The reference frequency will be distributed along the accelerator by a Rhode & Schwartz oscillator and the UTC time will be based on a Meridian II GPS, these 2 devices, to synchronize MRF cards, will be connected to the Event Master (EVM) card which is the main element of the timing system architecture.

The MRF timing system [1] thanks to an optical fiber network allows to distribute downstream and upstream events with a µs propagation time. Currently we are working on development to also use it for the machine protection system of the accelerator.

In this paper I will present the hardware used, timing architecture, developments and tests we have performed.

INTRODUCTION

SNRC and CEA collaborate to the upgrade of the SARAF accelerator to 5 mA CW 40 MeV deuteron and proton beams (Phase 2) at 176MHz. The timing system is a key part of the accelerator as the machine protection system. The CEA control team is in charge of them and we have selected the MRF products to provide an integrated solution.

TIMING SYSTEM ARCHITECTURE

Overview

The SARAF timing system main functionality is to distribute:

- Trigger events
- Timestamping
- Reference frequency

The main EVM is in charge of distributing these 3 above topics. This EVM generates the trigger events from sequencers, multiplexed counters or external inputs. The timestamping will be defined by the Meridian 2 GPS [2] that distributes the UTC time by NTP and the Pulse Per Second signal (PPS). The PPS consists of a signal sent to the EVM to increment the second of the timestamp with an accuracy of 10ns. The GPS also distributes a 10MHz with ultra-low phase noise to the master oscillator that distributes itself the reference frequency (176MHz) of the accelerator to the EVM and to the RF devices.

Propagation Time

For machine protection system or PostMortem analysis, we use the upstream propagation of events. For machine protection system, the event propagation time associated to a machine issue is an important point to be able to ensure that the system performance is suitable to the requirements. To test this feature, we built a test bench to measure the propagation of an event. The principle of the test is generating a signal on one input of an EVR, and measuring the time for the system to generate an output signal on any other EVR of the bench (see Fig. 1). On this test bench, the propagation time is about 700µs for each additional floor of EVM fan-out.

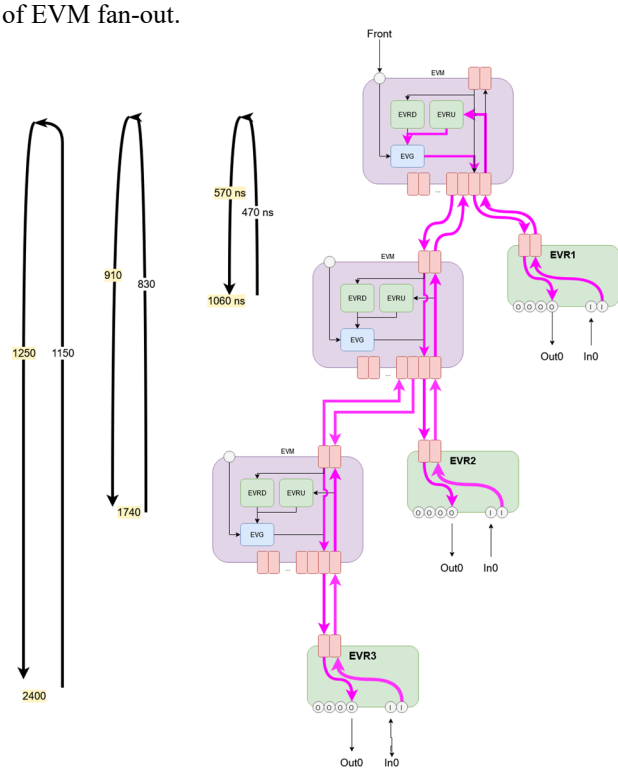


Figure 1: Test bench and result of event propagation.

Topology

The main topology is represented in Figure 2. Due to the propagation time described in the previous paragraph, we need to have as less as possible floors of EVM fan-outs. Therefore the distribution has been divided into 3 distinct parts: Injector, MEBT and SCL.

The Injector EVR has a direct link to the main EVM because it has an important role in protection and has to receive trigger events as soon as possible.

The MEBT (Medium Energy Beam Transport) part is controlled essentially by one EVM fan-out that will only be used as fan-out during beam operation. However for RF conditioning, it will produce events for RF to let cavities as independent as possible.

For the same reason in the SCL (Super Conducting Linac) part, each cryomodule will have a devoted EVM fan-out.

† alexis.gaget@cea.fr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV022

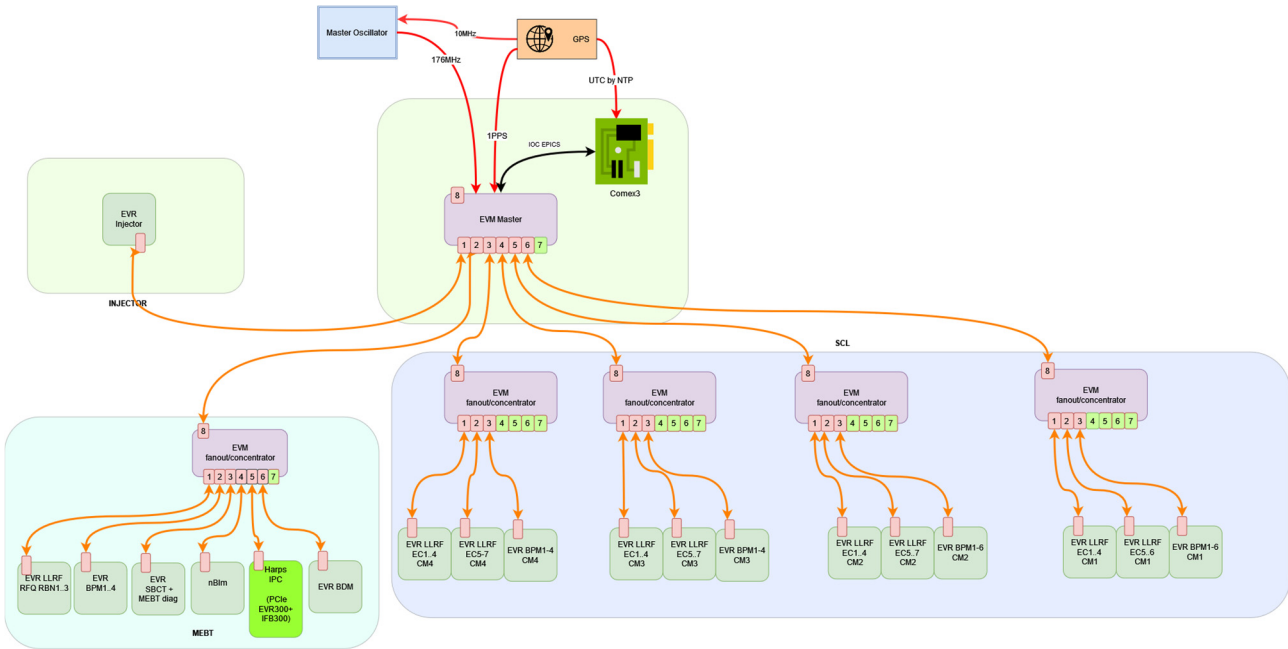


Figure 2: SARAF LINAC MRF Timing topology.

HARDWARE PLATFORM

SARAF that mainly rely on MTCA.4 platform with NAT R2 crate, NAT-MCH-PHYS80 and NAT-MCH-RTM-COMex-E3 [3], uses for the timing system the MTCA-EVM-300 and the MTCA-EVR-300U provided by Micro-Research Finland (MRF), which are already being deployed for the injector part. When an Industrial PC runs EPICS IOCS, the PCIe-EVR-300DC can be used. This EVR has a PCI express interface and can be plugged easily in a lot of computers. This EVR also has a micro SCPI interface to connect it on the IFB300 device offering up to 8 universal modules proposed by MRF to produce the synchronization trigger. Figure 3 shows an overview of the hardware.

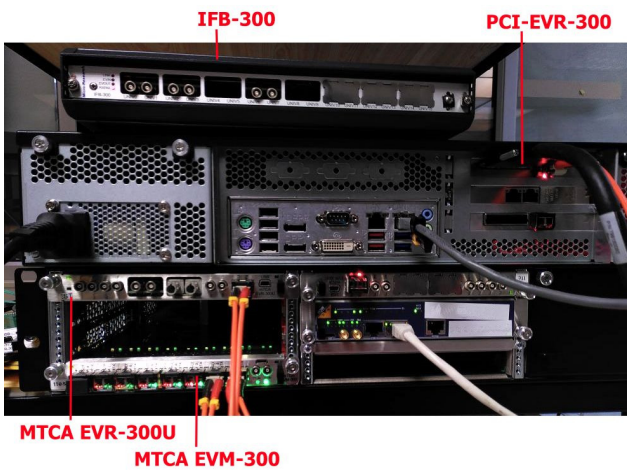


Figure 3: MRF hardware used in SARAF.

An EVM can be connected up to 7 MRF devices through optical fiber thanks to SFP connector. The 8th connector can only be used to get signals from an upstream EVM. This card can be used as the master, or as fan-out.

MTCA-EVR-300U has 4 LVTTL outputs, 2 TTL inputs and 2 slots to plug universal modules. Thanks to MTCA.4 this EVR can also use backplane lines of the MTCA.4 bus from the line 17 to 20 (see Fig. 4), meaning 8 channels can be configured as outputs or inputs.

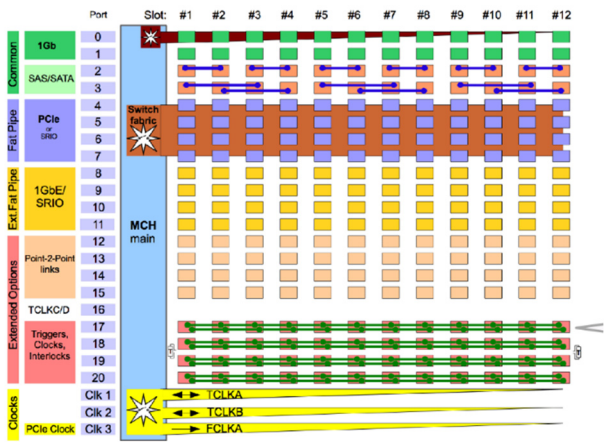


Figure 4: MTCA.4: MLVDS Bus.

Those backplane lines have been standardized for all the crates of the accelerator [4].

The Figure 5 shows the EVR design and its interface to the LLRF developed by Seven Solutions [5].

OPERATION MODE

Beam Operation

Beam operation is in charge of authorized or not the operator to perform actions on the accelerator like starting the source.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV022

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

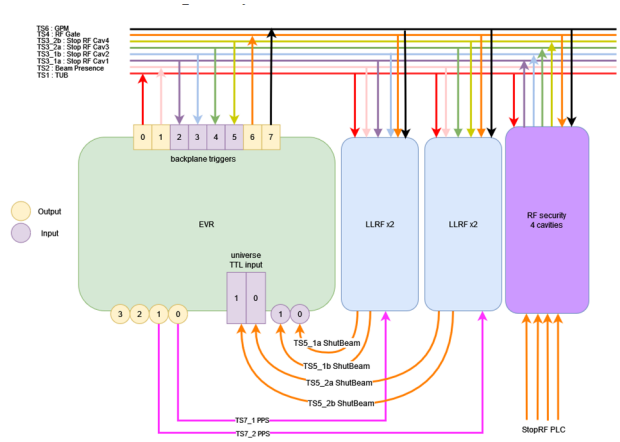


Figure 5: Distribution of signals on LLRF EVR.

For beam operation, we will use one of the sequencers of the EVM. The sequencer will trig events corresponding to the RF Gate, the chopper and the beam presence as shown in Figure 6.

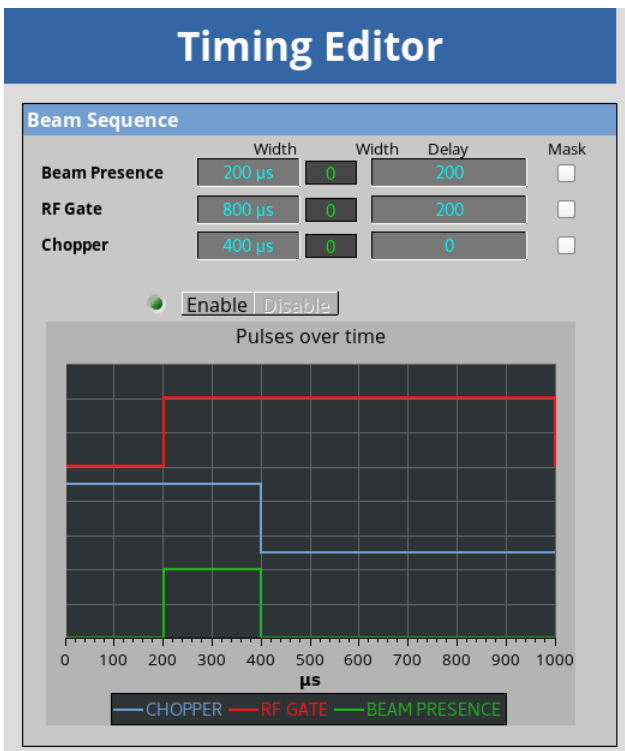


Figure 6: Graphical interface on CSS for the control of the sequencer.

We have different states during this operation:

- Pilot Beam: if the injection of beam is possible, the sequence is enabled and can send 3 different event triggers to manage the beam.
- Stop Beam: the sequence status is still on, but using the mechanism of mask of MRF, we mask the chopper event and then magnetron and RFQ are still on but the beam is not produced. It happens when we want to change the destination of the beam.
- Start Beam: the sequence status is still on and we unmask the chopper event trigger to generate the beam, it happens when the destination of the beam has been changed.
- Reset Beam: the sequence is stopped and no beam is produced.

RF Conditioning Operation

The sequence of the RF conditioning is handled by another EPICS application, but it uses the timing EPICS application to apply a new width to the RF Gate and to change the frequency period. As seen in Figure 7 in RF conditioning mode, the main EVM will not produce the trigger events, it's each sub EVM fan-out that will manage the RF trigger events for the conditioning of cavities. Meaning that each EVM will provide the same RF Gate for its cavities:

- EVM MEBT : RFQ and 3 rebunchers
- EVM CM1: 6 equipped cavities
- EVM CM2: 7 equipped cavities
- EVM CM3: 7 equipped cavities
- EVM CM4: 7 equipped cavities

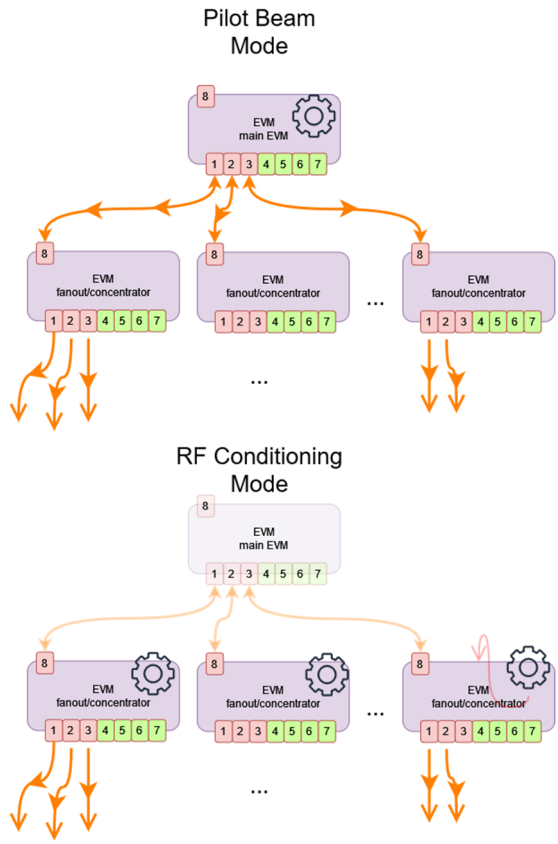


Figure 7: Timing distribution modes.

MACHINE PROTECTION SYSTEM

The machine protection system is the system that will shut down the beam and protect the accelerator from material damages. Based on ALBA accelerator experience [6], we've decided to delegate a part of the MPS to the MRF system. Thanks to the EVM-300, it's now easier to have an

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV022

upstream configuration. Indeed, the EVM integrates 2 internal EVRs (see Fig. 8) that can handle upstream events and associate the reaction of the EVM.

As seen in Figure 8, a signal detected on the input of an EVR can upstream to the EVM and be submitted to the EVRU (for EVR Upstream). From there, like any other EVR, the EVRU can be configured to act on the reception of an event and trig the event in the integrated EVG of the EVM and then redistribute events to the whole EVRs of the accelerator. We use among others this method to propagate a "shut beam" event to the whole accelerator when a critical default is detected by a device.

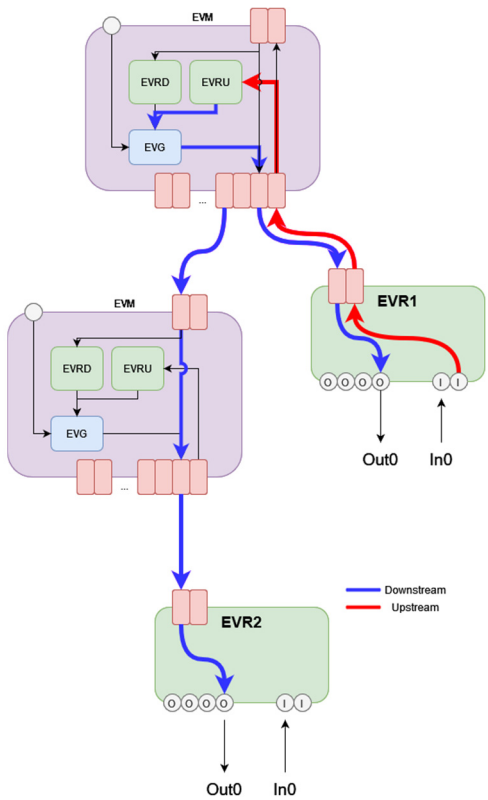


Figure 8: Upstream and propagation of an event to the whole MRF tree architecture.

EPICS APPLICATION

The EPICS timing application is based on mrfioc2 driver developed by Michael Davidsaver and a fork developed by ESS ICS [7]. We have modified the ESS driver to add mask functionalities of the sequencer [8]. We use the Irfu EPICS Environment (IEE) for EPICS development and deployment [9].

An SNL sequencer is implemented to handle the different Operation mode described earlier. Each EVR and EVM configuration will have an interface in order to operate simply the timing but an expert interface is also proposed for MRF developers. In the simplified interface (see Fig. 9), the user can modify the width and the delay of pulses produced by the EVR and have an overview of the status of pulses.

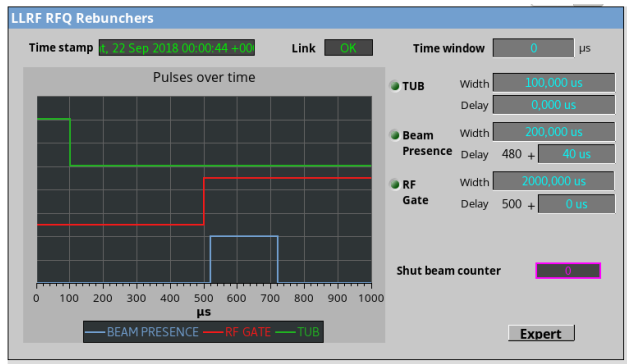


Figure 9: Example of simple interface for EVR LLRF.

CONCLUSION

We have completed the test of the MRF solution for the Timing and Machine Protection Systems at CEA Saclay in 2021. Due to the fact that those functionalities are transversal, they have been progressively installed at SOREQ this year.

ACKNOWLEDGEMENTS

The author would like to thank Jukka Pietarinen from MRF, Javier Cereijo Garcia and Jerzy Jamroz from ESS for their technical help.

REFERENCES

[1] MRF, http://www.mrf.fi

[2] Meridian 2 GPS, https://endruntechnologies.com/products/time-frequency/gps-frequency-standard

[3] NAT products, https://www.nateurope.com/products/board_level_products.html

[4] F. Gougnaud, "Status of The SARAF-Phase 2 Control System", presented at 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper MOPV001, this conference.

[5] J.S. Fernandez and P. Gil, "Status of the uTCA Digital LLRF design for SARAF Phase II", presented at 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, Oct. 2021, paper WEPV031, this conference.

[6] O. Matilla, D. Beltran, and D. Fernandez, "ALBA Timing System – A Known Architecture with Fast Interlock System Upgrade", in *Proc. 13th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'11)*, Grenoble, France, Oct. 2011, paper WEBPMS023, pp.1024-1027.

[7] ESS mrfioc2 driver, https://github.com/ic-shwi/mrfioc2

[8] IRFU mrfioc2 driver, https://github.com/aga-get/mrfioc2

[9] J.F. Denis *et al.*, "IRFU EPICS Environment", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New-York, USA, Oct. 2019, pp. 1172-1174. doi:10.18429/JACoW-ICALEPCS2019-WEPHA040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV025

A NEW TIMING SYSTEM FOR PETRA IV

T. Wilksen†, A. Aghababyan, K. Brede, H.T. Duhme, M. Fenner,
U. Hurdelbrink, H. Kay, H. Lippek, H. Schlarb,
Deutsches Elektronen-Synchrotron DESY, Germany

Abstract

The currently ongoing PETRA IV project at DESY proposes an upgrade of the PETRA III synchrotron light source towards a fourth-generation, low emittance machine. The goal is to provide X-ray energies in the regime of 10 keV bringing the electron beam production to its physical limits with respect to the smallest achievable source size, and thus approaching the diffraction limit.

The realization of this new, challenging machine implies a new design of the timing and synchronization system because requirements on beam quality and controls will become significantly more demanding with respect to the existing implementation at PETRA III. Furthermore, the PETRA IV baseline for the fast front-end electronics readout will be based on the MTCA.4 standard. Given the success of the at DESY developed MicroTCA.4-based timing system for the European XFEL accelerator, it has been chosen to utilize the MTCA.4 technology for the PETRA IV timing system as well.

We present in this paper general concepts of the timing and synchronization system, its integration into the control system as well as first design ideas and evaluations of the major timing system hardware component, a MicroTCA.4-based AMC.

THE PETRA IV PROJECT

The PETRA IV project [1] comprises the replacement of the existing 3rd generation synchrotron radiation source PETRA III with a state-of-the-art ultra-low emittance storage ring. This includes the upgrade of the storage ring infrastructure with a circumference of 2304 m as well as the redesign of the current pre-accelerator chain and construction of new beamlines. The 6 GeV storage ring will be operated at an RF frequency of 500 MHz as PETRA III. Currently, the PETRA III facility can run in either the timing mode with 40 equally distributed electron bunches at 100 mA stored beam or in brightness mode with 480 equally distributed bunches and 120 mA of beam current. The bunch pattern options are being under discussion since recent modifications of the lattice design allow now for patterns more similar to the PETRA III ones rather than what was proposed in the PETRA IV CDR.

The present booster synchrotron DESY II will be replaced by a new one, DESY IV, to meet the requirements of a low emittance beam injected into PETRA IV. While the LINAC section will be kept, the gun will be upgraded as well as the transfer section with the PIA accumulator has to be revised. The option of keeping DESY II for test beam production has to be taken into account when designing the timing and synchronization system even if this is not considered to be part of the baseline layout of the PETRA IV project. The entire facility is shown in Fig. 1.

Furthermore, the front-end electronics for diagnostics and instrumentation will be completely overhauled and based on the MTCA.4 standard. At the same time, it has been decided to change over the existing PETRA III control system using the TINE framework to a DOOCS-based one as used at the DESY FEL accelerators. This effectively leads to the necessity to redesign the entire timing and synchronization system and its controls for the storage ring as well as for the pre-accelerator chain.

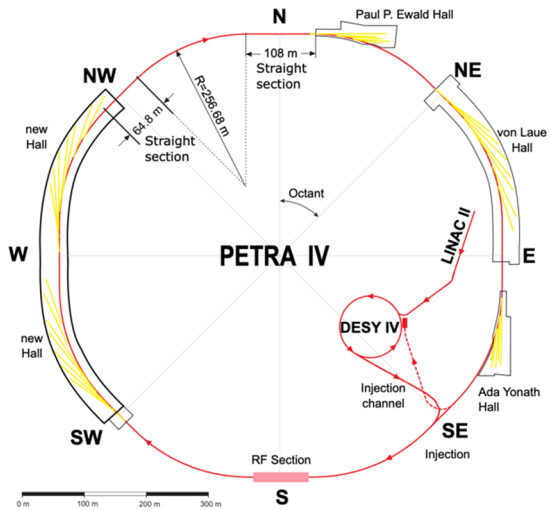


Figure 1: The PETRA IV facility layout with its existing, its new experimental halls and its pre-accelerators.

THE TIMING SYSTEM DESIGN

The overall concept of the new timing and synchronization system has to provide services to at minimum four individual synchrotron, booster, accumulator or LINAC facilities. Namely, the new PETRA IV storage ring, the new DESY IV booster ring, potentially a PIA II accumulator and a LINAC. If it is desired to continue to provide test beam services through DESY II fed by the same new LINAC and PIA II, one has to make sure, these are well enough synchronized with respect to the RF, timing and controls with the pre-accelerator chain for PETRA IV. The idea being pursued at the moment is to provide individual main oscillators and main timing system components for each of the major installations PETRA IV, DESY IV, DESY II and LINAC with PIA together. This would allow for a most flexible way of providing the different required RF frequencies and synchronization signals. Beamlines and experiments would be served by the timing system of PETRA IV, test beams by the DESY II one. The challenge

† Tim.Wilksen@desy.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV025

here will be to synchronize not just the RF reference signals but also the timing information itself, i.e. cycle numbers and such to uniquely tag the data across facilities for later analysis.

THE TIMING SYSTEM HARDWARE

Baseline

The hardware for the new timing and synchronization system will follow this baseline:

- MTCA.4 [2] will be used as the standard for the timing system electronics specifically for the transmitter and receiver module.
- Since MTCA.4 has been successfully introduced and established when constructing and commissioning the European XFEL accelerator timing system, the design of its transmitter and receiver Advanced Mezzanine Card (AMC) [3] will be used as a basis for the new PETRA IV MTCA.4 timing module.
- The design will be kept such that adaptions and modifications of the overall functionality can be implemented easily during the life-cycle of the accelerator.
- Essential to a successful realization is the expertise from both the machine beam controls and machine controls system groups, being the developers of the timing and synchronization systems for the FEL and synchrotrons at DESY.

Key Requirements

The key requirements for the timing and synchronization system of which the latter is crucial for the proper RF reference and signal distribution are given as follows:

- Distribution of a continuous RF reference signal
- Provision of low-jitter clocks for ADC sampling
- Distribution of timing signals, clocks and trigger events
- Provision of beam-synchronous data such as:
    - High-resolution timestamp
    - Revolution counter(s)
    - Beam modes i.e. timing or brilliance mode
    - Bunch pattern
    - Bunch current (PETRA IV)

Further, it has been proven at the EuXFEL that a common MTCA.4 module acting as a transmitter and/or as a receiver is a quite flexible and convenient concept.

A dedicated optical fiber network with drift compensation along the lines will be used. The topology follows the storage ring layout to make use of temperature-controlled media shafts and facilities. The storage ring, the booster and the LINAC with its gun will have its own timing system, however, being synchronized across the overall facility. Beamline experiments will have the possibility to connect to the PETRA IV timing system via the same MTCA.4-based AMC module.

Integration into The Control System

An important part of the PETRA IV control system will be to provide all relevant data not only to control and monitor the entire accelerator but also to analyze its performance and proper functioning. This requires the timing system to provide the corresponding functionality. Beam-synchronous information like a high-resolution timestamp, revolution counters and a bunch pattern might be utilized by:

- Post-mortem analysis
- Transient recorder
- Event archives
- Bunch-resolved data acquisition (for detailed analysis)
- Experiment controls and its beamline data acquisitions

PROJECT STATUS

Organization

The timing and synchronization system project is well embedded into the overall PETRA IV project management. A work-breakdown structure as well as a product-breakdown structure with all of its deliverables have been set up. The PBS represents a hierarchical view of all PETRA IV components:

- Define the scope “What will be built?”
- Organization of design and specification data
- Integration of the subsystem into the overall project

Requirements are derived for each PBS item and classified:

- Design, Interface, Functional
- Design can be validated against requirements
- Requirements are linked to PBS items

As of now requirements from accelerator subsystems, accelerator work packages as a well as beamline experiments are being collected, discussed and evaluated. This is an iterative process and will continue throughout the TDR stage. Eventually reviews will lead to a final design for the TDR documents.

MTCA.4 Hardware Design

Currently an evaluation phase is ongoing and accompanied by lab pre-tests. This is done with the existing x2timer AMC hardware as installed at the DESY FEL machines and shown in Fig. 2 but also with a potentially new hardware based on the MPSoC-enabled Xilinx Zynq technology. For this, a Xilinx Zynq evaluation board as shown in Fig. 3 is being evaluated.

The primary goal here is to test the clock and trigger generation function, as used in PETRA III, and to measure the jitter quality of generated clock and trigger signals at the output. Using a dual loop PLL clock cleaner is being investigated as an improvement to the output jitter.

At the same time components for the distribution of the incoming and outgoing clock and trigger signals on the AMC-Card are being researched and tested. Those components must have a very low additive jitter as well. Therefore, a test stand for measuring its signal jitter has been set up in the lab.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV025

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

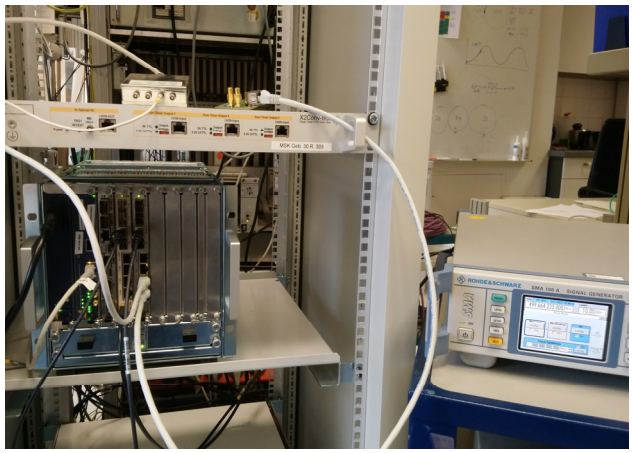


Figure 2: This figure represents the setup of the MTCA system with an x2timer AMC for tests and evaluation.

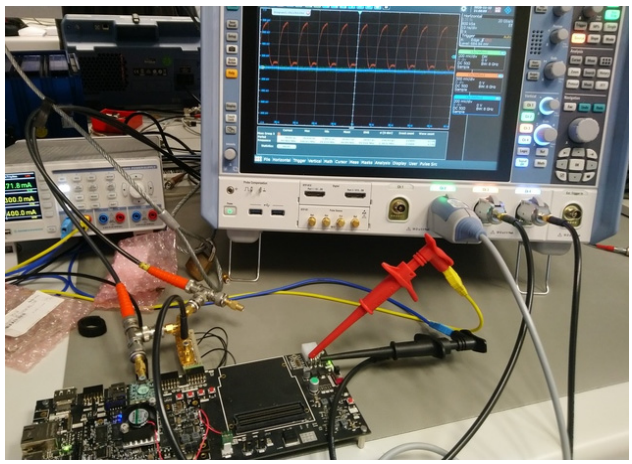


Figure 3: Shown in this picture is the measurement and FPGA evaluation board setup.

New concepts for the advance delay calculation which is required to target individual buckets inside the PETRA ring have to be developed as well as different kinds for the FPGA control system interface.

The next major step will be the integration of the aforementioned functionality into the existing DAMC-FMC1Z7IO AMC-Card. This MTCA.4 component is being developed by the DESY MSK group and the DESY MTCA Technology lab.

Since the overall distribution of the timing system information is using a distributed optical fiber topology, the clock recovery mechanism as well as drift compensation mechanisms have to be tested, too.

Eventually, the development has to yield in a design of a Zynq architecture-based successor AMC of the x2timer, currently being called x3timer. The design of this card is already in the specification phase and will be based on results of the requirements engineering in progress.

OUTLOOK AND ROADMAP

The next milestones within the TDR phase until mid of 2023 are:

- Collect requirements and derive specifications
- Develop MTCA AMC- and RTM-Card architecture and interfaces according to the specifications
- Adapt AMC- Design for x3timer demonstrator
- Hardware development and production
- Firmware development for the x3timer AMC
- Front-end controls development, i.e. server applications for controls

CONCLUSION

The renewal of the timing system for PETRA IV is without any choice however challenging. Given the excellent experience made with MTCA.4-based solutions specifically for the timing system hardware at the DESY FEL accelerators FLASH and the European XFEL, we are confident that this technology choice will serve the PETRA IV project quite well, too.

Our goal within the TDR phase is to fully specify the requirements of the PETRA IV timing and synchronization system and produce a fully functional demonstrator of the x3timer AMC hardware until 2023.

ACKNOWLEDGEMENTS

We acknowledge DESY (Hamburg, Germany), a member of the Helmholtz Association HGF, for its support on providing resources and infrastructure. Furthermore, we would like to thank all colleagues of the MSK and MCS groups as well as the PETRA IV project team and management for their contributions to this work and help in preparing this paper.

REFERENCES

[1] Schroer, C. G. *et al.*, "PETRA IV Upgrade of PETRA III to the Ultimate 3D X-ray Microscope Conceptual Design Report", Hamburg Deutsches Elektronen-Synchrotron DESY 2019, doi:10.3204/PUBDB-2019-03613

[2] The MicroTCA MTCA.4 Standard by PICMG, http://www.picmig.org/openstandards/microtca

[3] A. Hidvégi *et al.*, "Timing and triggering system for the European XFEL project - a double sized AMC board", 18th IEEE-NPSS Real Time Conference, Berkeley, CA, USA, June 2012, doi:10.1109/RTC.2012.6418093

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV027

APPLICATION OF THE WHITE RABBIT SYSTEM AT SuperKEKB

H. Kaji*, High Energy Accelerator Research Organization (KEK), Ibaraki, Japan
Y. Iitsuka, East Japan Institute of Technology (EJIT), Ibaraki, Japan

Abstract

We employ the White Rabbit system to satisfy the increasing requests from the SuperKEKB operations. The SuperKEKB-type slave node was developed based on the SPEC board and FMC-DIO card. The firmware was customized slightly to realize the SuperKEKB needs. The device/driver for EPICS was developed. The five slave nodes have been operated since the 2021 autumn run. The delivery of the beam permission signal from the central control building to the injector linac is taken care of by new slave nodes. The timing of the abort request signal and the trigger for the abort kicker magnet are recorded with the distributed TDC system. More slave nodes will be installed in the next year to enhance the role of the distributed TDC system.

INTRODUCTION

There are two types of modern timing systems for the large-scale accelerator. One is the event-base system. Its most famous hardware is Event Timing System (EVT) [1]. The other is the timestamp-base system. The famous hardware in this case is White Rabbit (WR) [2].

Both EVT and WR enhance their roles because of the increasing requirements from the accelerator operation. The synchronous control of the distant hardware is indispensable to improve the performance of accelerators.

The SuperKEKB collider [3] at KEK utilizes EVT for the injection control including the delivery of the timing-triggers towards the beamline. It successfully implements the transcendental scheme in the position injections [4, 5].

We consider introducing WR in addition to EVT for taking care of the increasing requirements to the SuperKEKB control system and to replace the devices which close with their lifetime. We especially plan to develop the distributed data acquisition system (distributed DAQ).

NETWORK-BASED CONTROL SYSTEM

In the operation of the modern accelerators, fast and robust communication between the separated hardware is indispensable. In such a case, the accelerator control is realized by synchronously operating modules that are connected via the dedicated optical cable. Often, the cable length becomes more than a kilometer in the large-scale accelerators. Recently, the roles of the timing system are increased for this purpose.

The network-based control system is summarized in Table 1. The module that is employed at SuperKEKB is listed together with the future possibilities of EVT and WR for those purposes.

* E-mail: hiroshi.kaji@kek.jp

Table 1: List of network-based control systems: the module that is employed at SuperKEKB is listed together with the future possibilities of EVT and WR. "Commercial" is a commercial product. "Original" means an original product that is developed at KEK.

	SuperKEKB	Possibility
Trigger delivery	EVT	EVT or WR
Bucket Selection	Commercial	WR
Abort system	Original	EVT or WR
Beam permittion	Original	WR
Distributed DAQ	None	EVT or WR

In the case of SuperKEKB, the trigger delivery for the beam injection is taken care of by EVT. The commercial distributed shared memory is utilized for Bucket Selection [6]. The original modules were individually developed for the Abort Trigger System [7] and the beam permission system.

On the other hand, the WR system has several kinds of modules so that they can take care of all listed systems. We consider replacing the "Commercial" and "Original" devices with WR. Then, all network-based control systems are developed with EVT and WR. Note, the best way is to develop everything with one system. However, to develop with two systems is still better. And this experience becomes important knowledge for future colliders like Higgs factory.

APPLICATION TO SuperKEKB

In this section, we report the WR application at SuperKEKB. The entire system is discussed after introducing the specification of the SuperKEKB-type slave node.

Slave Module

Figure 1 is the pictures of the slave modules which we developed at KEK. The SPEC board [8] is selected as the slave node of WR. Only the FMC-DIO card [9] is employed in the SueprKEKB operation, so far. Two types of nodes are produced at a low cost.

The module in the upper picture of Fig. 1 is a homebuilt computer with a 2.9 GHz 6 core CPU, 8 GB of RAM, and 2.5 inch SATA III SSD. We chose the small-size commercial barebone kit with the PCIexpress slot to mount the SPEC board. All parts are mass-produced goods and can be supplied easily. The size of this module is $20 \times 25 \times 8\,\mathrm{cm}^3$.

The firmware and software of the starting kit [10] are utilized with small customization. The transition timing of the input or output signals is informed in both higher or lower transitions while that is informed only in higher transition with the original firmware. The software runs on the Ubuntu 18.04 OS. We developed the device/driver for EPICS.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV027

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 1: Pictures of the slave node: the upper picture is SPEC on the homebuilt computer. In the lower picuture, SPEC is housed in the 19-inch box with 1U height. It can be operated from RaspberryPi.

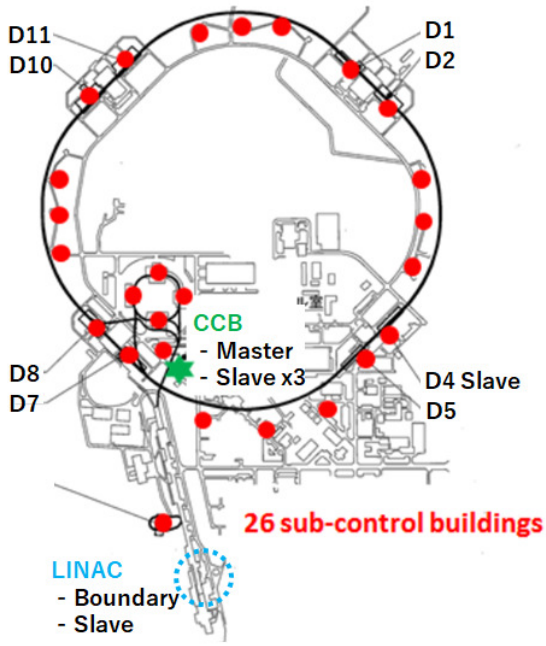


Figure 2: Map of the slave nodes at SuperKEKB: The WR modules were installed at D4, CCB, and LINAC buildings.

Followings are the functions of SuperKEKB-type slave node:

- Detect the transition of the TTL level signal for both input and output channels and inform the GPS synchronized timestamp.
- Transport the status of an input signal to the other nodes and enforce output.
- Output the TTL level signal according to the status of the received signal from other nodes.
- Output the pulse-per-second (PPS) signal which is synchronized with GPS.

The module in the lower picture of Fig. 1 is SPEC with RaspberryPi. In the 19-inch rack box, the SPEC board and the RaspberryPi 3 model B+ are assembled with the power supply units. This system utilizes the standalone feature of SPEC. The SPEC board is controlled via a mini USB port. It is successfully configured as the slave node. The test operation under the WR network at SuperKEKB was successfully carried out. However, the development of the EPICS device/driver is still ongoing.

Installation to Beamline

We installed the five SPEC nodes and two WR switches [11], so far. Their location is indicated in Fig. 2. Three SPEC nodes and one WR switch were installed at Central Control Building (CCB). For the WR switch as the grand-master module, the PPS signal and 10 MHz reference clock are provided from the rubidium clock which is synchronized with GPS. The slave nodes measure the arrival timings of the abort request signal at CCB and the delivery timing of the TTL triggers to the abort kicker magnet. The Beam Gate signal is input into one of the slave nodes. It is transferred to the injector linac (LINAC) [12]. Besides, the 99176 divided revolution signal (~ 1.002 Hz) is produced.

The SPEC node at the D4 building provides the PPS signal to the beam size monitor for the electron ring (X-ray monitor) [13]. Since the PPS signal from the WR slave node is synchronized with GPS, it indicates the absolute time of the measured data of the beam size monitor. The system is being tested in the 2021 autumn run.

Figure 3 is a picture of WR modules at LINAC. One WR switch is installed as the boundary module. One SPEC node is installed under this boundary module. It is utilized to receive the Beam Gate signal and delivers the TTL level control signal to the LINAC components such as the electron guns. In the future, more slave nodes will be installed under this boundary module to understand the LINAC hardware status with the common clock.

BEAM GATE SYSTEM "D0D0"

Beam Gate is a part of the beam permission system. The beam permission system permits or inhibits the provision of the beam pulse from LINAC. In general, the system monitors the accelerator hardware and inhibits the injection when some hardware is in an abnormal condition.

In the case of Beam Gate at SuperKEKB, the injection is enabled and disabled dynamically by the operator. It is managed to keep the beam current at two main rings (MR). The stored beam at MR is decayed and the operation beam current is decreased gradually. Beam Gate is opened and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV027

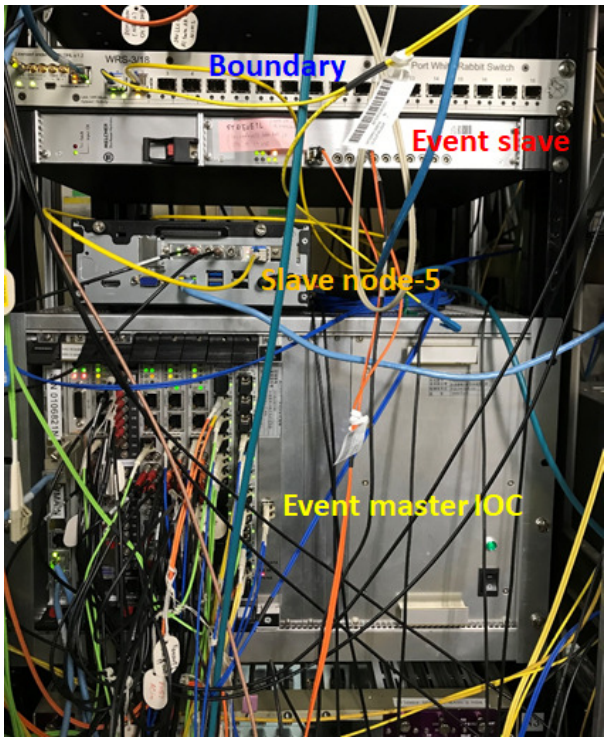


Figure 3: Picture of LINAC node: the WR switch and SPEC node were installed near the master IOC of EVT.

implements the beam injection when the operation beam current becomes the lower acceptable limit. Then, Beam Gate is closed when the operation beam current reached the target value of the experiment.

The beam permission signal is delivered from MR to LINAC via 800 m optical cable. It is inputted to the SPEC node at CCB and transferred to that at LINAC. This new system is named "D0D0". The name of the system is decided from the setting of the FMC-DIO card at LINAC[1].

One of the advantages of the D0D0 system is to know the accurate timing of both enabling and disabling the beam injection with GPS time. It is reliable information when we inspect the relation between the injection and problems which happened at MR.

The slave node can transfer the input signal to more than one node. So, in the future prospect, the Beam Gate signal can be delivered not only LINAC but the injection and extraction control of the damping ring (DR). In that case, the extraction system of DR must be controlled at least 40 ms later than the injection system. The signal transfer with the SuperKEKB-type node can take care of such kind of delay.

Besides, all beam permission logic can be included in the FPGA of SPEC. The Mock Turtle framework may uncomplicate the management.

[1] It is "**iD0D0**". Note, the setting of channels 2 and 4 is zero and is not the capital letter "O". However, we pronounce our system like "dóudou".

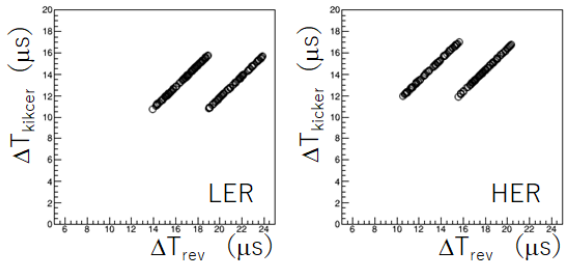


Figure 4: Measurement of correlation between abort response time, ΔT_{kicker}, and the revolution, ΔT_{rev}: the details of the setup of this measurement are explained in Ref. [15]. The results for both the positron and electron rings are shown in the left and right plots, respectively.

DISTRIBUTED TDC

The distributed Time-to-Digital Converter (distributed TDC) is one part of the distributed DAQ. The key technology of the distributed DAQ is the timestamp synchronization among the data acquisition devices which are distributed along the long beamline of accelerators. The utility of the timestamp synchronization among the separated beam monitors was demonstrated with the injection archiver of SuperKEKB [14].

The WR system is a strong tool to take care of timestamp synchronization. By using the SuperKEKB-type slave node as TDC, we can measure the timing of events that occur in the entire accelerator and can sort them in the time order with the accuracy of 8 ns.

We plan to measure the timestamp of the TTL reference-signal from the individual Abort Trigger modules [7] with the SuperKEKB-type slave node. The Abort Trigger System collects the request signal of the beam abort from the accelerator hardware and monitors. Therefore, we can make the time order of abort requests and know which component launches the abort request firstly. They are valuable data to diagnose the source of the problem. The time resolution of the abort request measurement will be 8 ns and becomes much precise than that in the current system[2]. We installed the slave node only at CCB, so far, and all modules will be installed in the next year.

The feasibility of the WR system in the abort timing measurement is demonstrated in Ref. [15]. In this case, only two modules were installed at the D2 building and CCB. Figure 4 is the update of the result with more large data set. The correlation between the abort kicker timing and the revolution was clearly observed. The variation of the abort response time can be explained by the timing of the abort gap.

[2] Note, the low accuracy part still remains in the upgrade system. The accuracy of the timestamp comparison among the channels on the same Abort Trigger module is 100 ns. It comes from the internal clock rate of 10 MHz.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV027

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

We are discussing further improvement. We plan to receive the signal directly from the beam loss monitor which has a high time resolution and will be installed soon.

TRIGGER DELIVERY

The trigger delivery from the WR node is the other valuable method to configure distributed DAQ. Since the WR node knows the precise timing of signal output, WR can fetch the GPS synchronized timestamp to data taken with its trigger. So far we can provide the following two kinds of triggers.

Pulse-to-Pulse

All slave nodes with SPEC and FMC-DIO can provide the PPS signal which is accurately synchronized with GPS. We provide the PPS signal to the X-ray monitor for the electron ring at the D4 building. It is the first step for this kind of distributed DAQ. The operation test is ongoing in the 2021 autumn run.

One-Herz Revolution

The 99176-divided revolution signal can be delivered from the WR slave node. Note the revolution of SuperKEKB is ~ 100 kHz. Therefore, the rate of this signal becomes ~ 1.002 Hz. The DAQ rate of some monitors at SuperKEKB is quite less than the revolution and is from 1 Hz to 10 Hz. This signal can use a such kind of case.

The signal is made from the 2254-divided revolution which is the source of the injection trigger. It is inputted into one of the WR slave node at CCB. Further 44 times division is implemented on the CPU process. The signal transfer to other nodes is implemened once in the 44 inputs while that process is vetoed in the remaining cases.

The trigger resolution in the current system is 8 ns. And there is a possibility of a future upgrade with FMC-TDC and FMC-DLY. In this case, the resolution of the revolution signal becomes less than a nanosecond. It will advance machine commissioning and the study of beam physics. For example, if we carry out the bunch-by-bunch measurement at SuperKEKB, the time resolution must be smaller than 4 ns.

CONCLUSION

The operation of WR has been started at SuperKEKB. The SPEC and FMC-DIO are employed as the slave node. By applying the small customization to firmware and software, the SuperKEKB-type slave node provides a lot of benefits to the accelerator operation. The Beam Gate system "D0D0" provides a more smooth operation of SuperKEKB. The distributed TDC system plays important role in the beam loss diagnostic.

The development of the distributed DAQ system with WR is continued and it will provide large benefits in the accelerator operations.

REFERENCES

[1] MRF website, http://www.mrf.fi/.

[2] White Rabbit website, https://white-rabbit.web.cern.ch/.

[3] Y. Ohnishi, *et al.*, "Accelerator Design at SuperKEKB", *Prog. Theor. Exp. Phys.*, vol 2013, no. 3, p.03A011, 2013.

[4] H. Kaji, "Bucket Selection for the SuperKEKB phase-3 operation", in *Proc. of 15th Annual Meeting of PASJ*, Nagaoka, 2018.

[5] H. Kaji, "Construction and Commissioning of Event Timing System at SuperKEKB", in *Proc. of IPAC'14*, Dresden, Germany, 2014.

[6] H. Kaji, "Bucket Selection for the SuperKEKB phase-3 operation", in *Proc. of 15th Annual Meeting of PASJ*, Nagaoka, 2018.

[7] S. Sasaki, *et al.*, "Development of Abort Trigger System for SuperKEKB", in *Proc. of 11th Annual Meeting of PASJ*, Aomori, 2014.

[8] Simple PCIexpress Carrier (SPEC) website, https://ohwr.org/project/spec/wikis/home

[9] FMC-DIO card website, https://ohwr.org/project/fmc-dio-5chttla/wikis/home

[10] White Rabbit starting kit website, https://ohwr.org/project/wr-starting-kit/wikis/home

[11] White Rabbit switch website: https://ohwr.org/project/white-rabbit/wikis/Switch

[12] M. Akemoto, *et al.*, "The KEKB Injector Linac", *Prog. Theor. Exp. Phys.*, vol. 2013, no. 3, p. 03A002, 2013.

[13] E. Mulyani, *et al.*, "First measurements of the vertical beam size with an X-ray beam size monitor in SuperKEKB rings", *Nucl. Instrum. Meth.*, vol. A, no. 919, pp. 1–15, 2019.

[14] H. Kaji *et al.*, "Archive System of Beam Injection Information at SuperKEKB", *J. Phys. Conf. Ser.*, vol. 1350, p. 012150, 2019.

[15] H. Kaji, Y. Iitsuka, "Performance Test and Initial Application of White Rabbit System at SuperKEKB", in *Proc. of 17th Annual Meeting of PASJ*, online, 2020.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV028

ANALYSIS OF AC LINE FLUCTUATION FOR TIMING SYSTEM AT KEK

D. Wang*, K. Furukawa, M. Satoh, H. Kaji, H. Sugimura, Y. Enomoto, F. Miyahara, KEK, Japan

Abstract

The timing system controls the injection procedure of the accelerator by performing signal synchronization and trigger delivery to the devices all over the installations at KEK. The trigger signals is usually generated at the same phase of an AC power line to reduce the unwanted variation of the beam quality. This requirement originates from the power supply systems. However, the AC line synchronization conflicts with the bucket selection process of SuperKEKB low energy ring (LER) which stores the positron beam. The positron beam is firstly injected into a damping ring (DR) to lower the emittance before entering desired RF bucket in LER. A long bucket selection cycle for DR and LER makes it difficult to coincide with AC line every injection pulse. This trouble is solved by grouping several injection pulses into various of injection sequences and manipulating the length of sequences to adjust the AC line arrival timing. Therefore, the timing system is sensitive to drastically AC line fluctuation. The failure of timing system caused by strong AC line fluctuation and solutions are introduced in this work.

INTRODUCTION

The electron/positron collider, SuperKEKB, is upgraded from the KEKB project since 2010 at KEK, whose goal is to update the world highest luminosity record and discover new particle physics by Belle II experiment [1]. The timing system at the 700-m long injector linear accelerator (LINAC) is responsible for the injection of all accelerator complex which consist of a 7 GeV electron high energy ring (HER), a 4 GeV positron low energy ring (LER), a 2.5 GeV Photon Factory (PF) and a 6.5 GeV PF-AR ring (see Fig. 1). During the phase-2 operation in 2018, a 1.1 GeV positron Damping Ring (DR) is constructed at the middle of LINAC to lower the positron beam emittance [2].

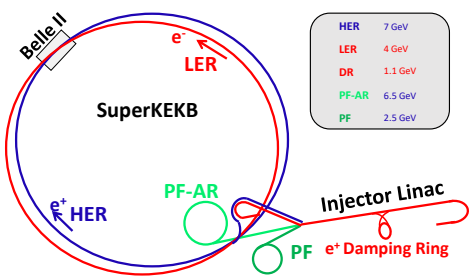


Figure 1: Overview of LINAC, SuperKEKB, and PF/PF-AR.

The event-based timing system at LINAC is required to switch the beam properties at 50 Hz by changing the event codes and additional control data. Totally 12 kinds of beam modes are defined to perform the pulse-to-pulse modulation. The master trigger signal of the timing system comes from the AC line to follow the fluctuation of power line and keep the beam energy stable. The bucket selection is implemented by adding delays to the master trigger signal. However, a long bucket selection cycle which caused by the DR makes it difficult to coincide with the AC line. A scheme called sequence shift is developed to synchronize with the 50 Hz AC line [3]. With the growth of the system complexity, several failures of timing system are observed and analyzed based on the fault diagnosis system [4].

In this paper, the reason of such failures are identified and some efforts to improve the reliability and stability of timing system are introduced.

BUCKET SELECTION FOR LER

The primary task for bucket selection is to provide the ability to select an arbitrary RF bucket in the ring for a single injection pulse. This can be achieved by adding a proper delay time to the gun triggering signal after defining the fiducial bucket in the ring. As the stable phase of RF cavity coincides based on the common frequency (CF) between LINAC and two main rings (MRs), the injection is performed based on the period of the CF [5]. After h times injection, all RF buckets are filled. The period during which all ring RF buckets can be filled is defined as a bucket selection cycle (BSC) and the period of BSC can be represented as

$$T_{\mathrm{BSC}} = h_{MR} * T_{CF} \tag{1}$$

where h is the harmonic number (i.e., the number of RF buckets), T_{CF} is the period of common frequency between LINAC and MR.

According to the requirements of RF synchronization, several significant frequencies for timing system can be calculated in Table 1. Note that the BSC for DR only is practically useful when performing DR-injection-only mode for beam study.

Table 1: Bucket Selection Frequencies at KEK LINAC

Frequency	Period	Remarks
2856 MHz	350 ps	RF frequency for LINAC
508.89 MHz	1.97 ns	RF frequency for DR & LER
114.24 MHz	8.75 ns	Event clock
2.21 MHz	452 ns	DR revolution frequency
99.39 kHz	10.06 µs	LER revolution frequency
45.15 kHz	22.15 µs	BSC for DR only
2.03 kHz	493 µs	BSC for LER only
88.19 Hz	11.34 ms	BSC for DR and LER
50 Hz	20 ms	Beam repetition rate

* sdcswd@post.kek.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV028

The harmonic number h_{MR} at two MRs are 5120, and the injection opportunity arises at the common frequency of 10.385 MHz between LINAC and the SuperKEKB MR (i.e., every 96.3 ns). According to the Table 1, all RF buckets can be selected within 493 µs, and the bucket selection process can be easily accomplished every 20 ms. Nevertheless, if DR is considered, the complexity of the bucket selection system grows. The harmonic number at DR is 230 and the least common multiple between the DR and LER harmonic numbers is 117760; thus, the BSC for DR and LER becomes 11.34 ms (i.e., 88 Hz).

AC LINE SYNCHRONIZATION

It is usually necessary to maintain the beam intensity and quality on the same level for consecutive triggers. Some devices are sensitive to the AC line phase and might not satisfy the stability requirement when working at different AC line phases. The klystron is a good illustration of this. Some klystrons use the AC-powered filament or heater to heat the electron cathode, and the changing AC phase might generate a magnetic field that eventually affects the velocity modulation process of the klystron. The frequency of the AC line is not ideal 50 Hz. The Tokyo Electric Power Company, which supplies AC power for KEK, adjusts the frequency of the AC power line within 50±0.2 Hz to balance the supply and demand requirement of the electricity market [6]. The timing system is required to follow the drift of AC50.

For HER injection, the BSC is 493 µs and the gun timing can be calculated based on the coincidence between AC50 and BSC. However, for LER injection with DR, the BSC is longer and to be 11.34 ms, so it cannot coincide with AC50 every 20 ms. Therefore, a method called sequence shift is utilized to deliver the triggers at the same phase of AC50, while it is also synchronized with the BSC.

SEQUENCE SHIFT

A set of injection pulses are grouped as an injection sequence. By shifting the length of the sequence, the synchronization between BSC and AC line becomes possible.

8/9 Pulses Sequence Shift

For HER injection, the BSC is 493 µs and the gun timing can be calculated based on the coincidence between AC50 and BSC every 20 ms. However, for the positron injection process, the bunch is first injected from LINAC into DR and then extracted from DR into LINAC after a pre-defined storage time. Hence, the BSC for the combination of DR and MR is 22.68 ms; therefore, synchronization with AC50 at every pulse is impossible. Consequently, an 8/9-pulse injection sequence was developed.

As shown in Fig. 2, the core ideology of the 8/9-pulse sequence is to define the 50 Hz fiducial point and record the AC50 arrival time and BSC timing. Inside the 8/9-pulse injection sequence, the bucket selection delay value for the positron can be acquired based on the fiducial point and current pulse number. After an 8-pulse sequence (i.e., 160 ms),

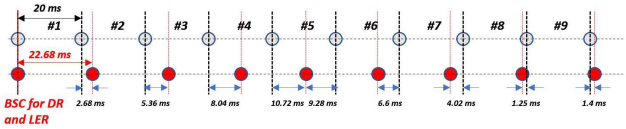


Figure 2: By shifting the injection sequence, the BSC for DR and MR injection can synchronize with the 50-Hz AC50.

the next sequence can synchronize with the BSC again if the next pulse starts 1.25 ms early. Similarly, launching the next pulse 1.4 ms after 9 pulses (180 ms) also meets the synchronization requirement (see Fig. 3). Using this method, the fiducial value for the BSC in every pulse can be calculated using the pulse number, and it is possible to operate the injection into DR and the extraction from DR at different pulses. The determination of the next sequence length is based on the AC50 value, which is measured by TDC for every pulse.

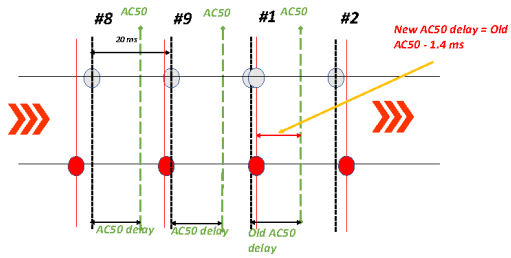


Figure 3: AC50 position in a pulse is manipulated by the sequence shift.

16/18 Pulses Sequence Shift

The DR operation is carried in 2018. To achieve a satisfactory damping effect, a DR storage time of at least 40 ms is required. There also exists a restriction of the maximum storage time in the DR. The injection and extraction delay are calculated together, and both follow the AC line phase at the injection pulse. Thus, the extraction timing trigger has a larger discrepancy at the downstream LINAC for a long storage time. The difference between the extraction timing and AC50 arrival timing at extraction pulse becomes larger when the storage time increases. Consequently, the maximum storage time is chosen to be 200 ms, while the minimum storage time is 40 ms. To supply the requirement of the maximum DR storage time (200 ms), the sequence length is doubled to apply the 16/18-pulse sequence shift in 2018, and the sequence shift values are enlarged to 2.5 and 2.8 ms [2].

Sequence Shift Algorithm

The algorithm of the sequence shift is based on the AC50 timing of the previous pulse and uses it as an estimation the AC50 value in the near future. A reference value is also required to act as a comparison point. Obviously, there are only two sequence types, i.e., 16-pulse sequence and 18-pulse sequence. Therefore, the AC50 shift value, i.e., 2.5

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV028

and -2.8, can be used to represent the sequence types 16 and 18, respectively. Because the program operation requires some time, the sequence shift decision should be done at the start of the current sequence. It is then easy to calculate the next sequence type based on the current sequence type and the AC50 timing of the starting pulse. The detail of the sequence shift algorithm can be referred from [7].

TIMING SYSTEM FAILURES

The timing system, which satisfies the requirements to operate the DR for the positron injection, was installed and commissioned in 2018 [2]. As the growth of the system complexity comes from the bucket selection and 16/18-pulse sequence shift, several failure modes of the timing system are observed. An event code log system, which monitors and saves all event codes received in EVR, is developed to diagnose the failure modes in 2019 [4]. The severity of some timing system failures is minimal because the trigger signals for devices are masked and inhibited by beam gate system [8] when some abnormal events are transmitted. On the other hand, a timing system failure that stops the delivery of event codes for a few minutes is severe for the physics run because it usually triggers the beam abort system to dump all the beams at SuperKEKB. It normally takes 10 minutes for the operator to check the status of the devices and restart the injection. The occurrence of such kind of failure is rare (i.e., 9 times) in 2020. However, the number of failures increases to 18 times in a month since the 2021 spring, and the beam operation is frequently interrupted. Thus, it is urgent and significant to understand the failure cause and stabilize the operation. By analyzing the event code log system and auxiliary timing data, such malfunction of the timing system is caused by strong AC50 drift.

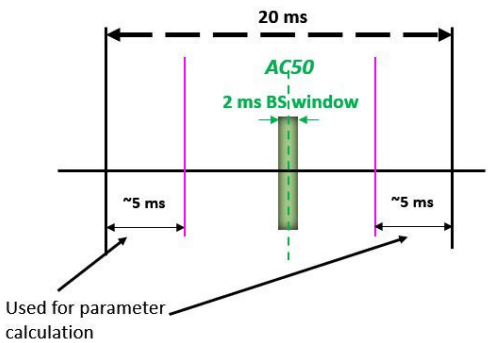


Figure 4: If AC50 arrives too early or too late in the 20 ms pulse, the logic of timing system may fail

As Fig. 4 shows, every pulse the timing system needs some time to process the data. If AC50 drifts continuously, the AC50 arrival time may comes too early or too late. Therefore, the timing system logic may fail.

STABILIZATION OF TIMING SYSTEM

As Fig. 5 shows, compared with the 16/18-pulse injection sequence, the 8/9-pulse injection sequence increases the robustness of the sequence shift algorithm. Switching to the 8/9-pulse injection sequence is a possible method to solve the timing failure because the AC50 adjustment speed is faster.

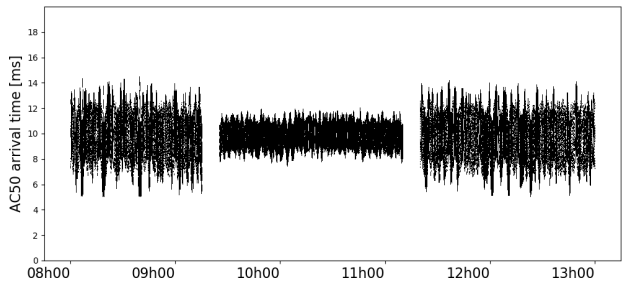


Figure 5: Comparison of AC50 arrival timing between 16/18-pulse sequence and 8/9-pulse sequence.

The restriction of the sequence length originates from the DR storage time and the root reason is the coupling of DR injection and extraction calculation. Shifting the RF phase at downstream LINAC is an efficient way to increase the DR extraction opportunity and relieve the stress of long sequence management. The DR extraction delay is calculated based on the shifted RF phase at downstream LINAC. Under these circumstances, the dependency between DR and MR injections can be removed. The operation of 8/9-pulse sequence shift is possible as the DR injection and extraction delay calculations can be separated. More descriptions about RF phase shifting can be found in [9].

Apart from the 8/9-pulse sequence shift, the sequence shift algorithm can be further improved by considering the AC50 drift value. Currently, the estimated AC50 arrival time is used to determine the next sequence type. If strong AC50 drift happens, the estimated value has a large deviation from the real value. The timing system could enter race conditions when an inappropriate sequence type is selected. To avoid such a situation, the sequence shift algorithm is modified and the average AC50 drift value in recent pulses is utilized to help the sequence shift algorithm to estimate the AC50 arrival time. The new estimation becomes closer to the real value even when AC50 fluctuates strongly.

According to our simulation, the new algorithm can handle an AC50 drift ranging from −156 µs to 158 µs. Figure 6 shows a timing system failure which is caused by the strongest AC50 drift situation that we have recorded during the past two years. The data was recorded by TDC in 2019. The AC50 drift value reached 120 µs and lasted for several seconds. Figure 7 compares three sequence shift schemes, 16/18-pulse sequence shift which is currently using, 8/9-pulse sequence shift, and 8/9 pulse sequence shift with the functionality of using the past AC drift value. The results show that the new algorithm can handle the most significant AC50 drift situation.

The 8/9-pulse sequence operation with AC50 drift compensation has been planned from the 2021 summer.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV028

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

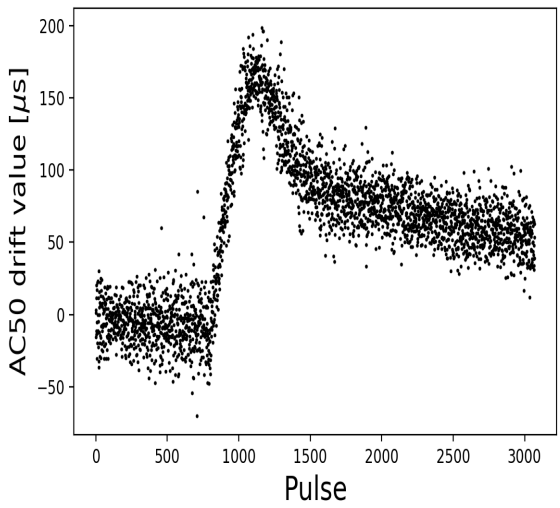


Figure 6: Extremely strong AC50 drift recorded by TDC.

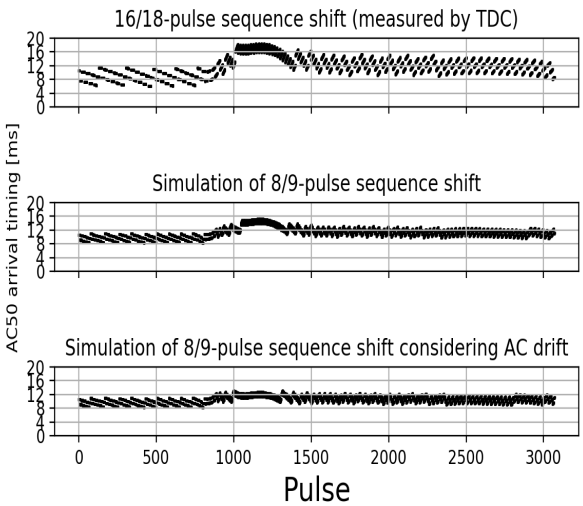


Figure 7: Comparison of different sequence shift schemes.

SUMMARY

The timing system failures that happened at LINAC frequently interrupt the operation of SuperKEKB. We analyzed the algorithm of the injection sequence shift, the timing system failures caused by strong AC50 fluctuation are solved after upgrading the sequence shift by applying short sequence length and performing a reliable AC50 estimation. The simulation results show that the new method can handle the most severe AC drift situation we have ever met in the last two years. The analysis procedure and solution for a timing system with a long bucket selection cycle can be referred to for the design of future circular accelerators.

REFERENCES

[1] Y. Ohnishi *et al.*, "Accelerator design at SuperKEKB," *Progress of Theoretical and Experimental Physics*, vol. 2013, no. 3, 26, 2013, doi:10.1093/ptep/pts083

[2] H. Kaji, H. Sugimura, Y. Iitsuka, and T. Kudou, "Injection Control System for the SuperKEKB phase-2 operation," in *Proceedings of the 15th Annual Meeting of Particle Accelerator Society of Japan*, 2018, pp. 124–128, https://www.pasj.jp/web_publish/pasj2018/proceedings/PDF/THOM/THOM03.pdf

[3] H. Kaji *et al.*, "Installation and commissioning of new event timing system for SuperKEKB," in *Proceedings of the 12th Annual Meeting of Particle Accelerator Society of Japan*, 5–Aug. 7, 2015, pp. 223–227, http://www.pasj.jp/web_publish/pasj2015/proceedings/PDF/FROL/FROL15.pdf

[4] D. Wang *et al.*, "The Fault Diagnosis of Event Timing System in SuperKEKB," in *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, 2019, pp. 741–745, doi:10.18429/JACOW-ICALEPCS2019-TUBPR04

[5] H. Kaji *et al.*, "Bucket Selection System for SuperKEKB," in *Proceedings of the 12th Annual Meeting of Particle Accelerator Society of Japan*, 5–Aug. 7, 2015, pp. 1278–1281, https://www.pasj.jp/web_publish/pasj2015/proceedings/PDF/THP1/THP100.pdf

[6] Tokyo Electric Power Grid Co., "Rule of frequency adjustment, supply and demand," 7, 2020, pp. 24–26, https://www.tepco.co.jp/pg/consignment/rule-tr-dis/pdf/freq-j.pdf

[7] D. Wang *et al.*, "Analysis and stabilization of AC line synchronized timing system for superKEKB," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1015, p. 165 766, 2021, doi:10.1016/j.nima.2021.165766

[8] H. Sugimura, H. Kaji, M. Satoh, F. Miyahara, and S. Sasaki, "Trigger control system with Beam Gate at SuperKEKB injector LINAC and Damping Ring," in *Proceedings of the 15th Annual Meeting of Particle Accelerator Society of Japan*, 7–Aug. 10, 2018, pp. 1078–1081, https://www.pasj.jp/web_publish/pasj2018/proceedings/PDF/THP0/THP091.pdf

[9] H. Kaji, "Bucket Selection for the SuperKEKB Phase-3 Operation," in *Proceedings of the 15th Annual Meeting of Particle Accelerator Society of Japan*, 2018, pp. 1114–1116, https://www.pasj.jp/web_publish/pasj2018/proceedings/PDF/THP1/THP100.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV029

DEVELOPMENT OF TIMING READ-BACK SYSTEM FOR STABLE OPERATION OF J-PARC

M. Yang†, N. Kamikubota, K.C. Sato, N. Kikuzawa, J-PARC Center, KEK&JAEA, Japan
Y. Tajima, Kanto Information Service, Tsuchiura, Japan

Abstract

Since 2006, the Japan Proton Accelerator Research Complex (J-PARC) timing system has been operated successfully. However, there were some unexpected trigger-failure events, typically missing trigger events, during the operation over 15 years. When a trigger-failure event occurred, it was often tough to find the one with the fault among many suspected modules. To solve the problem more easily, a unique device, triggered scaler, was developed for reading back accelerator signals.

The performance of the module has been evaluated in 2018. In 2021, we measured and observed an LLRF signal as the first signal of the read-back system for beam operation. After firmware upgrades of the module, some customized timing read-back systems were developed, and successfully demonstrated as coping strategies for past trigger-failure events. In addition, a future plan to apply the read-back system to other facilities is discussed. More details are given in the paper.

INTRODUCTION

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator complex. It consists of three accelerators: a 400-MeV H- Linac (LI), a 3-GeV Rapid Cycling Synchrotron (RCS), and a 30-GeV slow cycling Main Ring Synchrotron (MR) [1-2]. Since the initial beam in 2006, J-PARC has been improving beam power. Concerning MR, recent beam power is about 500-kW (50-kW) by fast (slow) extraction, respectively [3].

There are two time cycles used in J-PARC: A 25-Hz rapid cycle is used at LI and RCS, a slow cycle is used at MR. When MR delivers proton beams to the NU and HD facilities, 2.48-s (fast extraction mode (FX)) and 5.20-s (slow extraction mode (SX)) cycles are used, respectively. Because the slow cycle determines the overall time behavior of the accelerators, it is also called a "machine cycle."

The control system for J-PARC accelerators was developed using the Experimental Physics and Industrial Control System (EPICS) framework [4]. In addition, a dedicated timing system has been developed [5-6]. The J-PARC timing system consists of one transmitter module and approximately 200 receiver modules. Both types of modules were developed as in-house VME modules. Event-codes, which have information on beam destination and beam parameters, are distributed from the transmitter module to the receiver modules. A fiber-optic cable network is used for event-code distribution using several optical-to-electrical (O/E) or electrical-to-optical (E/O) modules. According to the received event-code, each receiver module generates eight independent delayed trigger signals.

Since the first beam use began in 2006, the J-PARC timing system has contributed to a stable operation of the accelerator beam [6]. Nevertheless, some timing trigger-failure events have occurred during beam operation. During each recovery process against a failure, it was often difficult to find a definite module among the many modules suspected. Such experiences have prompted us to develop a new module that can read back signals generated by the J-PARC timing system. We developed a new module, called a triggered scaler module, for this purpose [7-8].

In this paper, using one of trigger-failure events occurred in J-PARC MR as a case, the working principle, a usage in beam operation, and firmware upgrades of the triggered scaler module, are described. Moreover, a customized read-back system is introduced, followed by a discussion on the plan for future.

PAST TRIGGER-FAILURE EVENTS

Since 2006, we experienced some unexpected trigger-failure events during beam operation [8-9]. Herein, one case is given as an example, a 25-Hz irregular trigger event.

From November to December 2016, an O/E module, which was used to send a 25-Hz trigger signal from RCS to MR, started to produce irregular triggers (Fig. 1). Because the irregular triggers affected a critical beam diagnostic system, the accelerator operation was suspended several times per day [7]. It took 2 weeks to identify the troublesome O/E module.

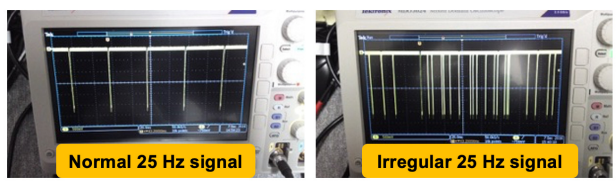


Figure 1: The normal and irregular 25-Hz trigger signals monitored by an oscilloscope during beam operation.

TRIGGERED SCALER MODULE

Introduction

A unique device, triggered scaler (hereafter TS), was designed by J-PARC control group for reading back timing signals. It is a scaler to count number of pulses in a specified accelerator cycle, and it stores the counts in a momentary array [7-8]. The differences between a TS module, a digitizer, and a simple scaler are shown in Fig. 2. Contrast to the simple scaler, the uniqueness of the triggered scaler

† yangmin@post.kek.jp

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV029

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

module is that it has an external trigger input and an output data as a digitizer, and its output is a data buffer with 192 elements. It was designed as a Yokogawa (FA-M3/e-RT3) PLC-type, which can fit the standard I/O form in J-PARC MR.

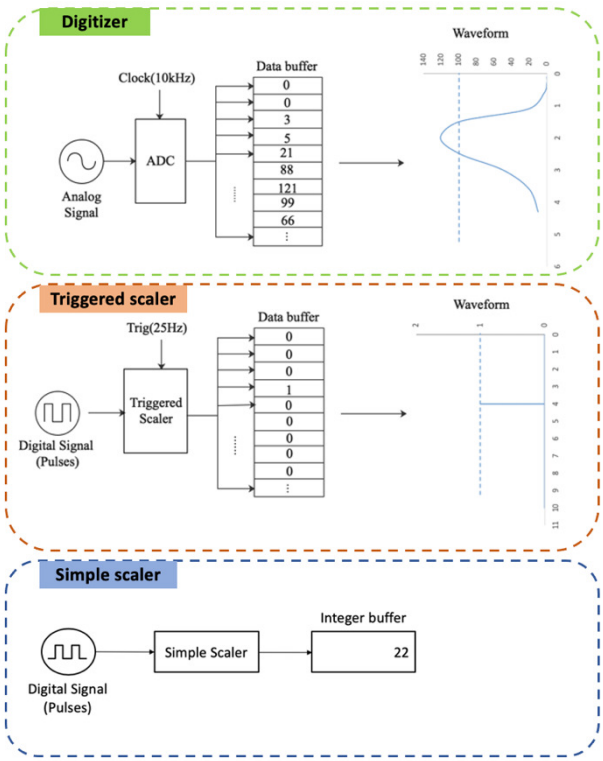


Figure 2: The comparison of digitizer, triggered scaler, and simple scaler.

There are two types of the module: the MR-type and the LI-type. The hardware appearances of the two types are shown in Fig. 3. The LI-type TS module was designed to find a missing trigger event of the LI accelerator. The module checks the pulse count in the rapid cycle (25-Hz), and outputs an error signal when an error is detected. In turn, the MR-type checks the pulse counts in the last slow cycle (2.48-s or 5.20-s) and raises an error flag. Both require the start signals of the slow cycle ("S") and the rapid cycle ("Trig"), provided by the J-PARC timing system. The MR-type has four input channels and no output channel, while the LI-type has two input channels and two error-output channels. There is a dual ring buffer (192 cells × 2 for MR-type, 448 cells × 2 for LI-type, 16 bit/cell) for each channel.

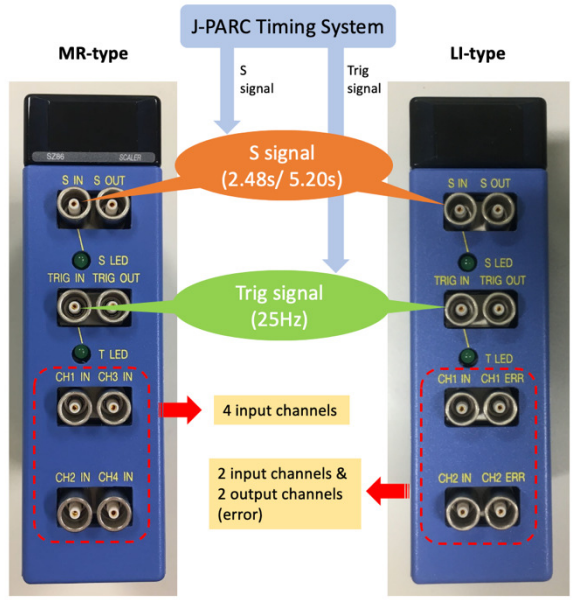


Figure 3: The hardware appearances of two types of the triggered scaler module.

Working Principle

In principle, each channel of the TS module works as a scaler. Two internal FPGA logics are shown in Fig. 4. The first logic (FPGA_1) counts the input pulses. When the "S" signal arrives, FPGA_1 starts to increase the count in the first cell of the first memory buffer. Each time the "Trig" signal arrives, FPGA_1 shifts the pointer to the next cell. When the following "S" signal arrives, the pointer is moved to the first cell of the second memory buffer. In other words, each cell keeps several trigger pulses received in a 40-ms bin (as the "Trig" signal is 25-Hz). This scheme enables retrieval of the numbers of counts during the last machine cycle. The second logic (FPGA_2) reads the memory buffers, checks whether trigger faults exist or not, and outputs an error signal for LI-type or sets an error flag for the MR-type, if necessary. The LI-type module can output an error signal directly through the output channels.

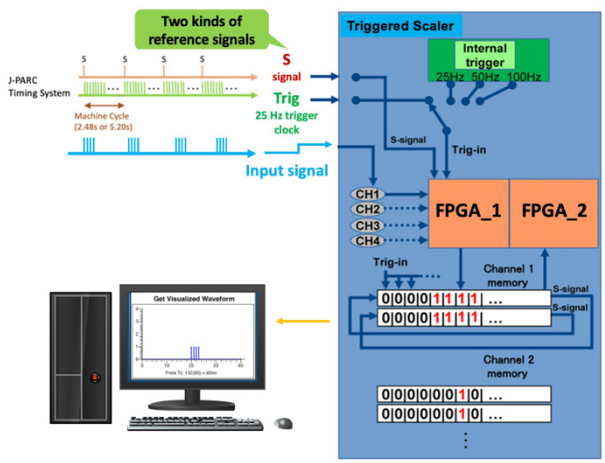


Figure 4: The conceptual design of the triggered scaler module.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV029

For Beam Operation

The performance of the module was confirmed and reported in 2018 [7]. Early in 2021, a setup for beam operation was prepared as shown in Fig. 5. It consists of a CPU module, a TS module, and a power supply module. All of them are standard Yokogawa PLC modules. Linux and EPICS IOC are running on the CPU module.

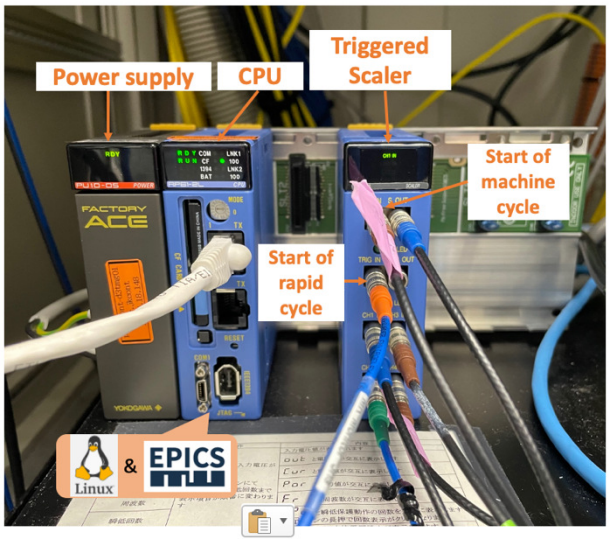


Figure 5: Test setup of triggered scaler module.

In May 2021, we started to observe an LLRF signal (MR circulation signal) by the setup, as one of the first operational surveillance to read-back signals. LLRF patterns in 30-GeV and in 8-GeV are visualized as shown in Fig. 6 (a). An EPICS PV (Process Variable) for the MR energy, which is deduced from the observed LLRF patterns, was developed and has been recorded by an archiver (as shown in Fig. 6 (b)).

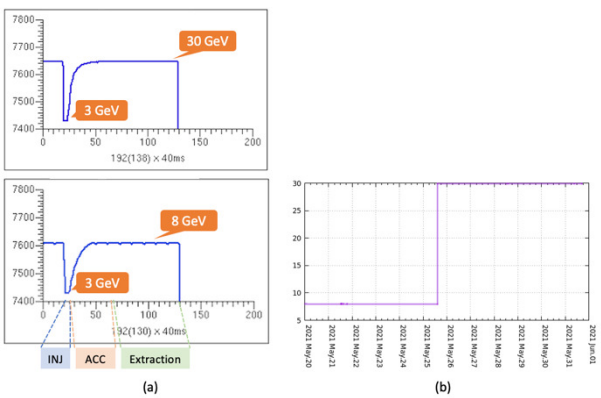


Figure 6: (a) Measurement of an LLRF signal, (b) MR energy from an archiver.

Firmware Upgrades

Based on experiences of module measurements, we identified small problems in the TS module. In 2020 and 2021, we upgraded the firmware of the MR-type module as follows (details are given in [9]).

1. Synchronize the internal trigger with the "S" signal. In the old firmware, the cell position of the received signal sometimes shifted to the next position.
2. Add preset-delay function for "Trig" signal. This function is to adjust the arrival time of the "S" signal relative to the external "Trig" signal.
3. Add 30-Hz, 60-Hz, and 120-Hz internal triggers to use in western Japan and other countries.
4. Clear data buffer each time when "S" signal comes.
5. A bug in counting the last cell of the data buffer was fixed. This bug has caused zero count in the last cells (62nd or 130th) when we had measured the RF signal.
6. Enlarge the LED flushing duration up to 10-ms, for better visibility in field works.

The issues 1 and 5 were caused by unsynchronized internal trigger. With the new firmware, the internal trigger is synchronized with the "S" signal. The issues 2 and 3 are essential upgrades to use the module in other facilities. The issues 4 and 6 were updated for user's conveniences.

This firmware upgrade was significant to shift the module status from a test to an operational phase. The firmware upgrade of the LI-type module is planned.

CUSTOMIZED TIMING READ-BACK SYSTEM

It has been demonstrated that the TS module can measure and visualize accelerator signals with the relationship of accelerator cycle. Several customized timing read-back systems were developed based on the module, as the countermeasures against the past trigger-failure events [9]. Here shows one example: a read-back system of 25-Hz trigger.

The 25-Hz trigger from RCS is important for the beam diagnostic system of a fast-current transformer (FCT), which observes the in-coming protons into MR. The past trigger-failure event in 2016, as shown in Fig. 1, caused a critical problem.

The read-back system of 25-Hz trigger observes the signal at the MR side since May 2021.The system realizes remote monitoring of the signal as shown in Fig. 7. Thus, we can find the same failure event immediately.

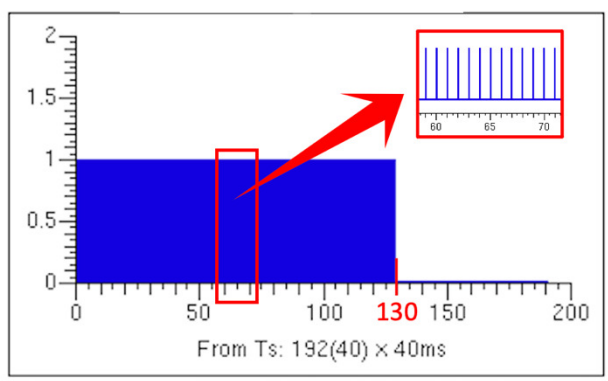


Figure 7: The observed 25-Hz trigger signal from RCS in a machine cycle (5.20-s) by the read-back system.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV029

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

FUTURE PLAN

The current read-back system assumes the use of the computer resources of the J-PARC control system. Thus, it is unavailable elsewhere, but on the J-PARC site. We would like to develop a standalone read-back system, which is available for other accelerators.

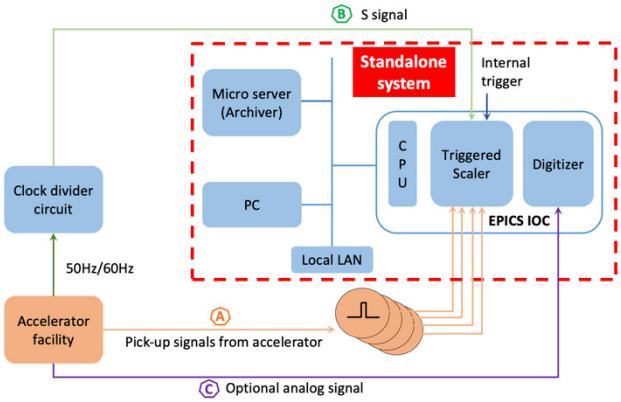


Figure 8: A standalone read-back system for other accelerators.

Fig. 8 shows a planned design of standalone read-back system. The system requires the start signal "S" from the target facility, which is needed to start the TS module. If there is no "S" (or similar) signal, a clock divider circuit would be used to simulate the signal from 50-Hz/60-Hz. The internal trigger of the TS module is to be used. Thus, no external trigger clock is needed. With additional I/O modules like a digitizer, the system can meet to various cases in other accelerator facilities. The unexpected trigger-failure detection of the target timing signal could be customized by updating the EPICS IOC software upon request. A portable micro-server is expected to work as an archiver to analyse faulty events.

If the portable standalone read-back system is successfully developed, it can be exported to other facilities.

CONCLUSION

A triggered scaler module was developed to read back the signals of the J-PARC timing system. The module is capable of detecting accelerator signals as well as visualizing beam based on the relationship with the accelerator cycle. The first read-back system by the TS setup started operational surveillance for an LLRF signal. Then, a customized read-back system for the 25-Hz trigger is introduced. It is demonstrated as a countermeasure against the trigger-failure event in 2016.

An idea to develop a standalone read-back system has been discussed. We expect the development of the standalone system will contribute to the accelerator timing fields in the future.

ACKNOWLEDGEMENTS

We express our best thanks to Noboru Yamamoto and other J-PARC accelerator staff members for their encouragement and suggestions during system development.

REFERENCES

[1] J-PARC website, http://j-parc.jp/index-e.html

[2] S. Nagamiya, "Introduction to J-PARC", *Prog. Theor. Exp. Phys*., vol. 2012, no. 1, p. 02B001, Oct. 2012. doi:10.1093/ptep/pts025

[3] S. Igarashi *et al.*, "Challenging to higher beam power in J-PARC: achieved performance and future prospects", in *Proc. 10th Int. Particle Accelerator Conf. (IPAC'19)*, Melbourne, Australia, May 2019, pp. 6-11. doi: 10.18429/JACoW-IPAC2019-MOYPLM1

[4] N. Kamikubota, *et al.*, "J-PARC control toward future reliable operation", in *Proc. ICALEPCS2011*, Grenoble, France, 2011, paper MOPMS026, pp. 378-381.

[5] F. Tamura, *et al.*, "J-PARC timing system", in *Proc. ICALEPCS2003*, Gyeongju, Korea, 2003, pp. 247-249.

[6] N. Kamikubota, *et al.*, "Operation status of J-PARC timing system and future plan", in *Proc. ICALEPCS2015*, Melbourne, Australia, Oct. 2015, pp. 988-991. doi:10.18429/JA-CoW-ICALEPCS2015-WEPGF121

[7] N. Kamikubota, *et al.*, "Development of triggered scaler to detect miss-trigger", in *Proc. PCaPAC2018*, Hsinchu, Taiwan, 2018, pp. 213-215. doi:10.18429/JACoW-PCaPAC2018-THP21

[8] M. Yang, *et al.*, "Applications of triggered scaler module for accelerator timing", in *Proc. 22nd Virtual IEEE Real Time Conference*, 2020. arXiv:2010.14716v2 [physics.acc-ph]

[9] M. Yang, "Development of Timing Read-Back System toward Stable Accelerator Operation", Ph.D. thesis, The Graduate University for Advanced Studies, SOKENDAI, Japan, 2021.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV031

UPGRADE OF TIMING SYSTEM AT HZDR ELBE FACILITY

Ž. Oven, L. Krmpotić, U. Legat, U. Rojec, Cosylab, Ljubljana, Slovenia
M. Kuntzsch, A. Schwarz, K. Zenker, M. Justus, Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany

Abstract

The ELBE center for high power radiation sources is operating an electron linear accelerator to generate various secondary radiation like neutrons, positrons, intense THz and IR pulses and Bremsstrahlung. The timing system, that is currently in operation, has been modified and extended in the last two decades to enable new experiments. At the moment parts of this timing system are using obsolete components which makes maintenance a very challenging endeavour.

To make the ELBE timing system again a more homogenous system, that will allow for easier adaption to new and more complex trigger patterns, an upgrade based on Micro Research Finland (MRF) hardware platform is currently in progress. This upgrade will enable parallel operation of two electron sources and subsequent kickers to serve multiple end stations at the same time. Selected hardware enables low jitter emission of timing patterns and a long-term delay compensation of the distribution network. We are currently in the final phase of development and with plans for commissioning to be completed in 2022.

SYSTEM DESCRIPTION

Hardware

The new timing system uses hardware from Micro Research Finland [1]. MRF hardware allows a modular approach with highly flexible topology with event master modules (EVMs), which are responsible for timing event generation and distribution, and event receiver modules (EVRs), which are responsible for setting the state of a physical outputs based on the received event.

For the majority of the system hardware used shall be of the MicroTCA form factor wherever possible, but certain user/experiment stations will use PCIe cards, which allow for up to 10 trigger outputs in a smaller package and are more cost effective.

Each of the two ELBE injectors will have an EVM, which will be responsible for generating independent timing pattern, allowing for both independent operation in separate beamlines and also common emission into ELBE accelerator.

Beam diagnostics, low level frequency control (LLRF) and other devices that need coordinated triggering will be connected to the EVRs via MicroTCA backplane trigger lines (if device lives in the same crate as EVR and supports triggering on a backplane signal), front panel output (if TTL level is needed) or through any of the Universal Modules that provide variety of optical and electrical level triggers.

As the number of outputs is limited to 8 per EVR, 4 front panel TTL trigger signals and 4 signals from Universal Modules, MRF offers to extend the number of trigger signals via rear transition module (RTM), where 10 additional trigger output signals can be configured and routed. Type of the input or output signal from the RTM is determined by the type of the Universal Input/Output (I/O) module selected as shown in Figure 1.

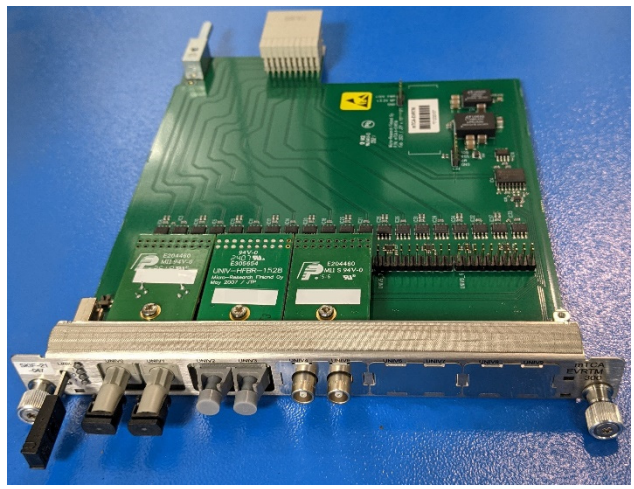


Figure 1: Micro Research Finland timing receiver rear transition module equipped with universal output modules.

All MRF boards (EVMs and EVRs) from the 300 series provide a delay compensation feature, where event propagation delay through whole distribution network is continuously measured and EVRs adjust their internal delay to match a programmed desired target delay. Each of the EVRs can have a different target delay setpoint, but adjusting target delay on an EVR will hold you back for approximately 15 minutes while the measurement and the loopback mechanism stabilizes.

To synchronize new timing system with the accelerator RF we plan to lock the internal oscillator of the master EVM to an externally provided frequency of 130 MHz through the dedicated input on front panel of the master EVM. This external reference frequency will be locked to both 1.3 GHz RF frequency and 26 MHz thermionic gun master oscillator and it will provide a 130 MHz event clock rate with which timing events will be distributed from the EVMs to the EVRs.

ELBE has multiple beamlines (as shown at Figure 2) but at the moment only one kicker is used to split the beam into two beamlines. For each beamline branch a dedicated bunch pattern has to be generated and sent to appropriate gun. A combined interlock and logic block interleaves the pulse trains and allows the machine interlock system to disable individual gun clock signals. As an upgrade to the machine protection system (MPS), an upgrade is foreseen where it will be able to react on interlock events after the kicker and only disable the corresponding pulse train.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV031

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

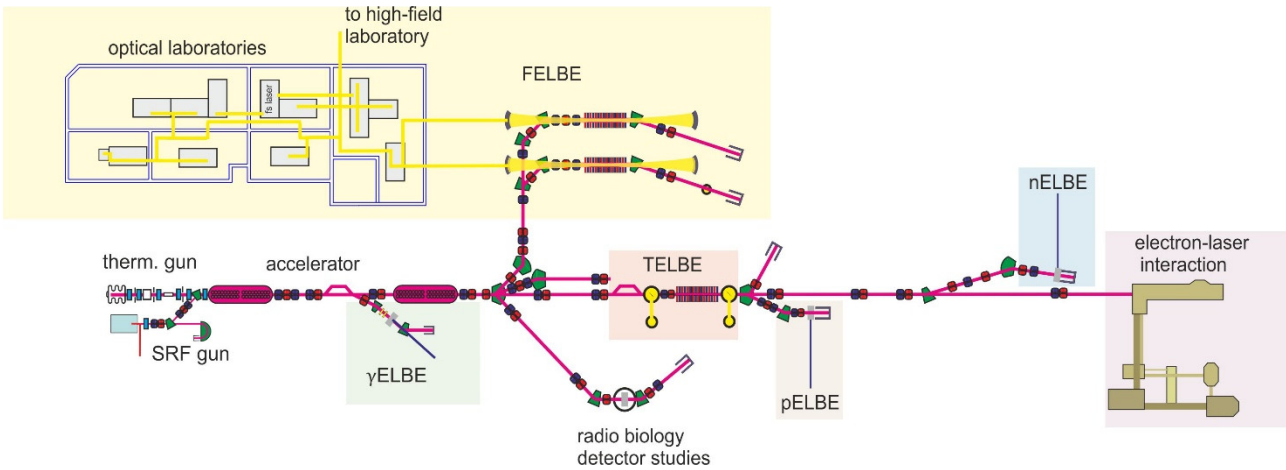


Figure 2: Topology of ELBE beamlines.

A possible ELBE upgrade scenario is to extend capabilities for simultaneous multi-user experiments; new timing system has a requirement to support parallel injection of the two electron sources. Selected timing hardware already has these capabilities. To make use of this feature to its full potential, injector sections of ELBE have to be modified to be susceptible for beams at different energy.

Software

Software for the new timing hardware is implemented in EPICS [2] control system framework with support of the mrfioc2 [3] and s7plc [4] modules. It allows a creation of required timing event patterns based on a requested operation mode, like continues wave (CW), macro pulsed or single pulsed beam. These predefined operation modes simplify the operation of the timing system for the end users, and allow for high level of automated parameter settings which is the core part of development effort.

The timing system can be configured using two graphical user interfaces (GUI). While EPICS GUI is mainly for expert control and debugging, the main control will be done by WinCC SCADA system which includes all ELBE machine controls.

All user entered machine parameters and operation mode dependent derivatives are checked by verification part of the timing software and by a master programmable logic controller (PLC) which is the main interface for MPS.

While in development a simulator written in Python is used to test connection and interaction between the timing system software and master PLC.

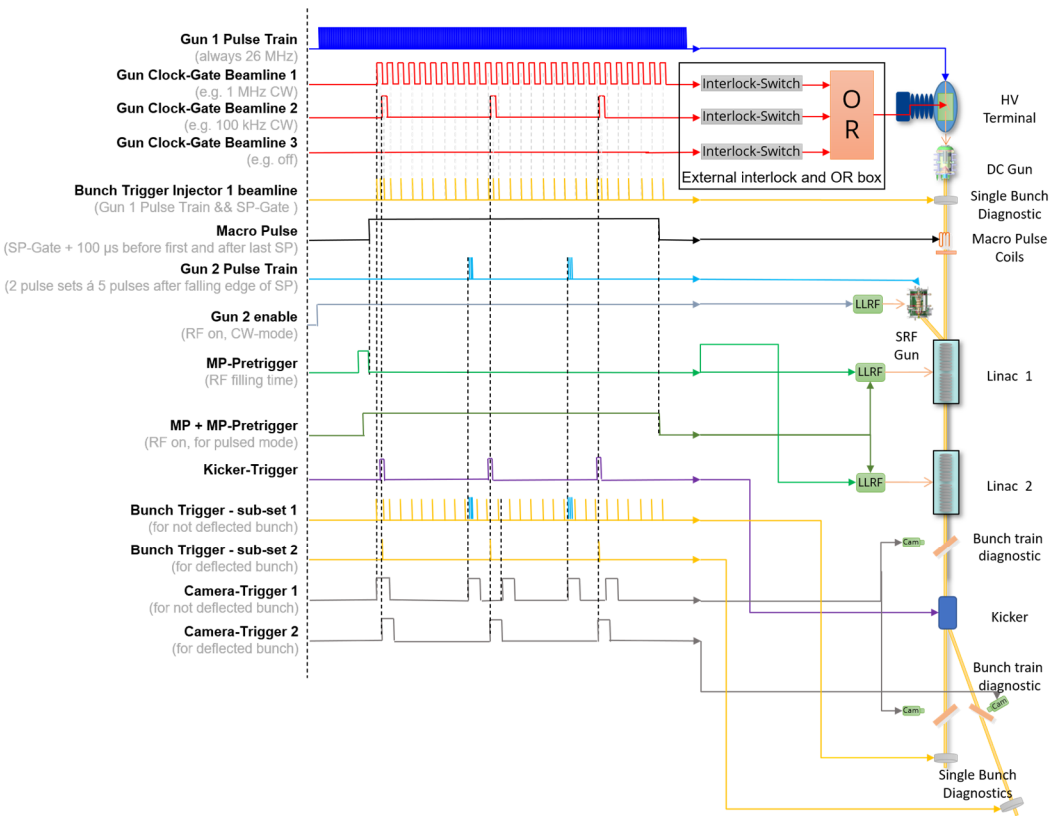


Figure 3: Example of a timing structure at ELBE using two guns and a subsequent kicker to generate bunch pattern.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV031

Figure 4: Output performance of a UNIV-LVPECL-DLY output module.

Pattern Generation - Example

Figure 3 illustrates an example of a more complex operation mode than could be applied after the timing system upgrade. Here both injectors emit electron pulses into the ELBE accelerator. A kicker is used to split the beam of the first gun into two beamlines. For each beamline section the corresponding gun emission triggers are generated and later the single bunch diagnostic triggers (e.g., for BPMs and ICTs) as well as the bunch train diagnostics (e.g., cameras).

The phase between the emission triggers can be set manually while the system automatically avoids gun emission triggers in the same RF bucket.

Output Jitter Performance

The output jitter was measured with two different MRF Universal I/O modules, one with UNIV-LVTTL and second with UNIV-LVPECL-DLY module using a Rohde & Schwarz FSWP8.

Measurements with UNIV-TTL module show jitter of 4.5 ps rms in an interval of 1 Hz to 1 MHz, while measurements with UNIV-LVPECL-DLY module show jitter of 7.8 ps rms in an interval of 1 Hz to 1 MHz as shown in Figure 4.

WORK IN PROGRESS

The system is currently under development on a test bench. It consists of two MicroTCA chassis equipped with two EVMs and four EVRs. In addition, a computer system is set up to test the PCIe event receiver modules. The generated patterns are analysed by three oscilloscopes.

The last of the main operation modes (dark current suppression, combination of macro pulse and single pulse) is being implemented and debugged at the moment. For stand-alone operation a GUI in Control System Studio GUI has been developed over many iterations and will serve as an expert GUI when the system has been commissioned at ELBE.

Main part of developed GUI is shown in Figure 5 and it enables the user to set up the timing system parameters through set of EPICS variables and also check what parameters are configured through WinCC GUI and under which what parameters timing hardware is currently operating. It also enables configuration of individual outputs on EVRs with selection of the predefined logic pulse patterns or with a pattern based on individual event, delay and width.

Status of the individual timing system hardware components is monitored and also displayed in dedicated GUI.

In addition to the implementation of operation modes the hardware is being tested and the long-term stability monitored.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV031

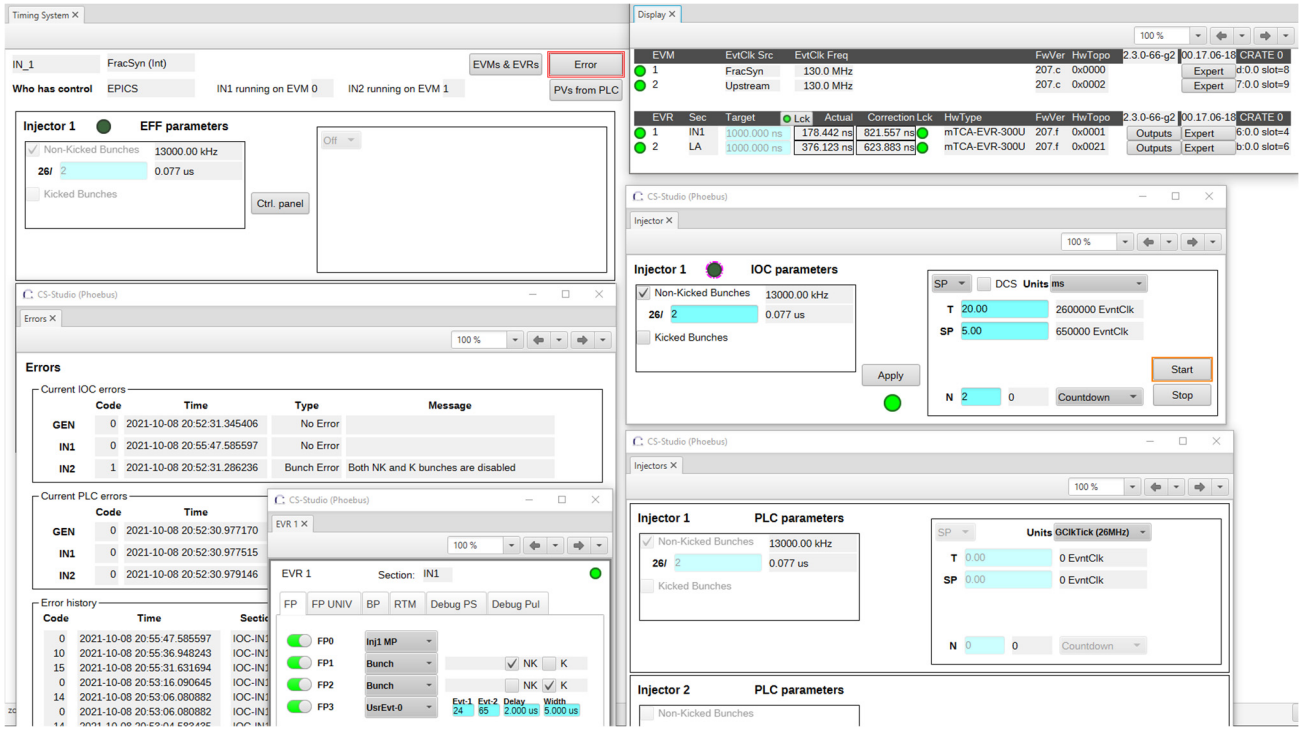


Figure 5: Timing system expert GUI.

CONCLUSION

A new flexible timing system is being developed for the ELBE accelerator. The system allows flexible bunch pattern generation for two injectors operated in parallel.

The system is currently set up on a test bench which is used for performance tests and debugging. First tests with the ELBE accelerator are planned for beginning of next year which will then follow by a stepwise transition from the legacy timing system to the new timing generation and distribution.

REFERENCES

[1] Micro Research Finland Oy, http://mrf.fi

[2] EPICS – Experimental Physics and Industrial Controls System, https://epics-controls.org

[3] EPICS module mrfioc2, https://github.com/epics-modules/mrfioc2

[4] EPICS module s7plc, https://github.com/paulscherrerinstitute/s7plc

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV032

THE DEMONSTRATOR OF THE HL-LHC ATLAS TILE CALORIMETER

Pavle Tsotskolauri†, on behalf of the ATLAS Tile Calorimeter System*
Tbilisi State University, Tbilisi, Georgia

Abstract

The High Luminosity Large Hadron Collider (HL-LHC) has motivated R&D to upgrade the ATLAS Tile Calorimeter. The new system consists on an optimized analogue design engineered with selected radiation-tolerant COTS and redundancy layers to avoid single points of failure. The design will provide better timing, improved energy resolution, lower noise and less sensitivity to out-of-time pileup. Multiple types of FPGAs, CERN custom rad-hard ASICs (GBTx), and multi-Gbps optical links are used to distribute LHC timing, read out fully digital data of the whole TileCal, transmit timing and calibrated energy per cell to the Trigger system at 40 MHz, and provide triggered data at 1 MHz. To test the upgraded electronics in real ATLAS conditions, a hybrid demonstrator prototype module containing the new calorimeter module electronics, but still compatible with TileCal legacy system was tested in ATLAS during 2019-2021. An upgraded version of the demonstrator with finalized HL-LHC electronics is being assembled to be tested in testbeam campaigns at the Super Proton Syncrotron (SPS) at CERN. We present current status and results for the different tests done with the upgraded demonstrator system.

Introduction

The upgrade of the Large Hadronic Collider (LHC) to the High-Luminosity Large Hadronic Collider (HL-LHC) is aimed to deliver up to ten times peak luminosity [1]. HL-LHC is designed to deliver collisions at the luminosity of 7.5×10^{34} cm^{-2} s^{-1} and up to 200 simultaneous proton-proton interactions per bunch crossing. This environment necessitated a Phase-II upgrade of the ATLAS detector [2]. The Tile Calorimeter (TileCal) is the central section of the hadronic calorimeter of ATLAS. It plays an important role in the measurements of jet and missing transverse momentum, jet substructure, electron isolation, energy reconstruction and triggering, including muon information. To meet the requirements of HL-LHC, upgraded electronics were tested by using The Hybrid Demonstrator in real conditions. The Hybrid Demonstrator combining fully functional upgraded Phase-II electronics with analog trigger signals to be compatible with present and legacy ATLAS interface. Demonstrator comprises four prototype mini-drawers, each equipped with 12 Photo-Multiplier Tubes (PMT) with 3-in-1 cards, Mainboard, Daughterboard and high voltage regulation board. Finger Low Voltage Power supply (fLVPS) is powering all four mini-drawers with 10V. The Hybrid Demonstrator is connected to an off-detector Pre-Processor module with a patch panel. PreProcessor modules provide data and control interfaces between on-detector electronics and both legacy and Phase-II Trigger and Data AcQuisitions interface (TDAQi) [3].

Photo-Multiplier Tubes

At every bunch crossing light produced by scintillator plates is transmitted by wavelength shifting fibres. PMTs are responsible for converting this light coming from TileCal cells into analog signal and transfer it to the next stage of a signal chain. Every PMT is equipped with a High Voltage Active Divider (HVAD). The function of HVAD is to divide high voltage coming from high voltage system to 8 PMT dynodes. The high voltage for PMTs is in the range of 600-900 Volts. HVAD is also responsible for linear PMT response. PMT Block consists of PMT, HVAD and 3-in-1 card. For individual PMTs and PMT Blocks, there are two different test benches to ensure their performance and correct functionality. Before PMT Blocks are assembled each PMT is tested to ensure their physical properties. After this PMT Blocks are tested with Portable Readout Module for Tile Electronics (PROMETEO) system which ensures the correct functionality of PMT blocks alongside other demonstrator modules [3].

3-in-1 Card

The 3-in-1 card is part of the PMT Blocks, see Figure 1. They are responsible for shaping, amplification and integration of signal coming from PMT. 3-in-1 cards feature a 16-bit dynamic range, 50ns full width at half maximum (FWHM) time constant, fast readout with two gains (low gain and high gain), integrated slow readout, charge injection for continuous calibration over full dynamic range. Low-gain signals are summed into trigger towers (adder cards) and sent to off-detector electronics. It also consists of an analog trigger to be compatible with the current ATLAS architecture. Data from 3-in-1 cards are read by FPGAs from The Mainboard.

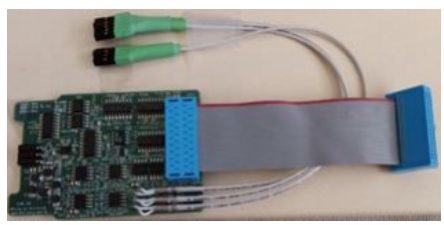


Figure 1: 3-in-1 Card [3].

Mainboard

The Demonstrator consists of four mainboards. The picture of The Mainboard is shown in Figure 2. Currently, Mainboard went through four revisions. The Mainboard is responsible for data transfer between PMT Blocks and Daughterboard. Each of them is connected to 12 PMT Blocks using Field Programmable Gate Arrays (FPGA).

† pavle.tsotskolauri@cern.ch
* Copyright [2021] CERN for the benefit of the ATLAS Collaboration. Reproduction of this article or parts of it is allowed as specified in the CC-BY-4.0 license

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Each FPGA reads out 3 PMT Blocks. For The PMT Block readout, The Mainboard uses 12-bit ADC at 40 Msps and 16-bit ADC for slow integration. The Mainboard is designed to be reliable and redundant, therefore it is divided into two sections called A- and B- Side. Each section is completely independent with each side having +10V LVPS supply bricks. In ordered to prevent failure caused by LVPS or total failure each side is connected with a "Diode-Or" connection to make the power supply more redundant. Other functions of The Mainboard are to provide timing signals for low- and high-gain Charge Injection (CIS) calibrations. Mainboard V4 is the final revision that will be used in HL-LHC.

Figure 2: Mainboard [3].

Daughterboard

The Daughterboard is an on-detector communication board that interfaces between front-end electronics and back-end Tile Pre-Processor (TilePPr) module, The Daughterboard is shown in Figure 3. This is 4th version of The Daughterboard. It sends PMT data and Detector Control System (DCS) and detector readout data to TilePPr over multi-gigabit links. Like The Mainboard, The Daughterboard is also divided into two sections for reliability and redundancy. Redundant optical fibres are also used for protection against single link failure. CERN-developed GBTx protocol chip is used for FPGA configuration, LHC clock distribution and to receive DCS and run commands from off detector PreProcessor module. The Daughterboard version 6 is available and is being tested at CERN test beam facilities. Replacement with the latest version on The Demonstrator is under consideration.

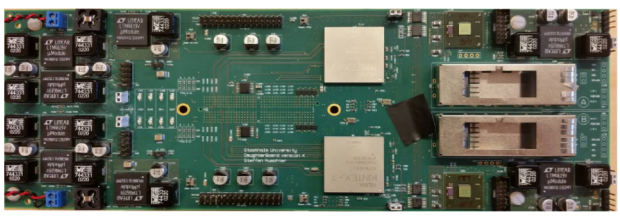


Figure 3: Daughterboard [3].

PreProcessor

The PreProcessor (PPr) is a core module of off-detector electronics. PMT Digital samples are transferred to The PPr every bunch crossing, A picture of the PPr is shown in Figure 4. The PPr has bi-directional communication with front-end electronics. It provides DCS commands, Timing, Trigger and Control information and the LHC Clock to the front-end electronics. From front-end electronics, The PPr receives PMT data which is stored in pipeline buffers waiting for trigger decision. In parallel, The PPr provides reconstructed energy information to the trigger system at 40MHz. After the trigger decision, The PPr sends buffered data to legacy Read-Out Driver (ROD) and processes in the same way as for current modules (backward compatibility).

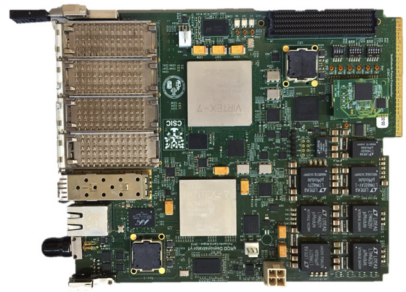


Figure 4: Tile PreProcessor [3].

Trigger and Data AcQuisition interface

Trigger and Data AcQuisition interface (TDAQi) communicates with off-detector electronics. It is responsible for constructing the trigger primitives and interfaces with trigger systems. The Trigger FPGA calculates trigger objects such as jets, muons, electrons with different granularity and energy resolution and results are transmitted through low latency optical links to the first level of the trigger. TDAQi is designed for HL-LHC and currently is not used for Demonstrator.

Low Voltage Power Supply

The Tile Calorimeter Low Voltage Power Supply (LVPS) provides power to all front-end electronics and provides control and feedback to the monitoring system and can be seen in Figure 5. Previous LVPS modules were generating eight different voltages for various sub-circuits of front-end electronics. In Phase-II upgrade each super-drawer is powered by one LVPS module which now provides 10V supply to all the front-end. As the result, TileCal has 256 LVPS boxes. Each LVPS box consists of eight bricks which coverts 200V input voltage to 10V output. For redundancy eight bricks are grouped into four sets of two. Diode-Or design is also implemented for improved reliability.

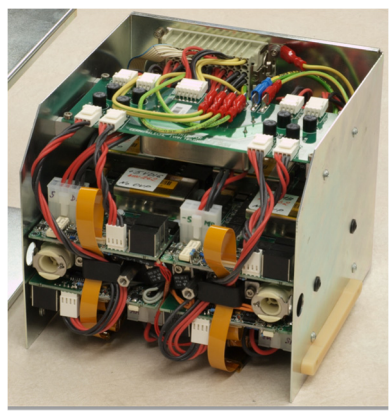


Figure 5: Low Voltage Power Supply [3].

High Voltage Power Supply

High Voltage Power Supply (HVPS) must supply high voltage to all PMTs in the system. it also needs to monitor, control and report values to DCS System, as shown in Figure 6. There is a total of 256 high voltage regulation boards each equipped with an ethernet interface.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV032

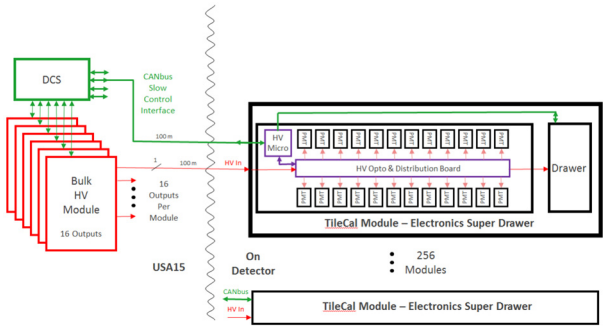


Figure 6: High Voltage Power Supply System [3].

Test Beam Results

The TileCal modules equipped with The Hybrid Demonstrator, with Phase-II upgrade electronics together with modules equipped with the old electronics, were exposed to different particles and energies in 7 test beam campaigns at CERN SPS North Area, during 2015-2018 [3].

The uniformity of the test module, which is segmented into three longitudinal layers (A - Figure 7 BC - Figure 8 and D - Figure 9), was evaluated using muon beams.

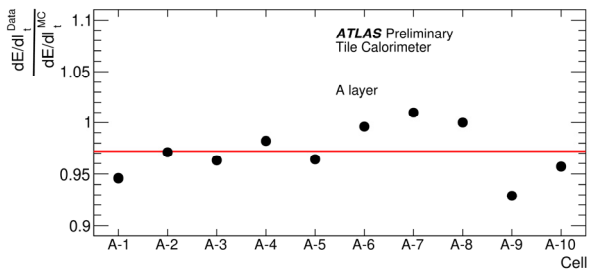


Figure 7: Longitudinal layer A [4].

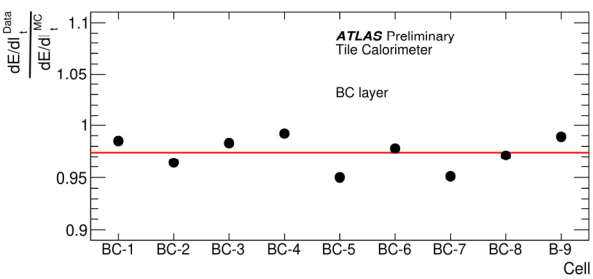


Figure 8: Longitudinal layer BC [4].

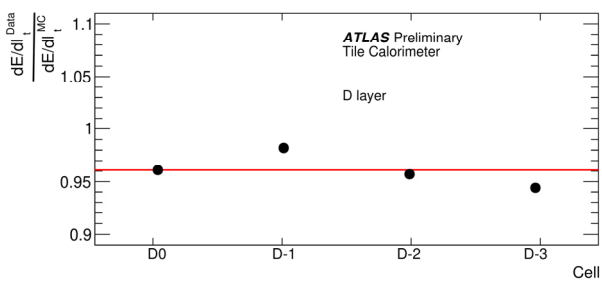


Figure 9: Longitudinal layer D [4].

Figure 10 shows the distribution of the total energy deposited in the calorimeter obtained using experimental and simulated electron data.

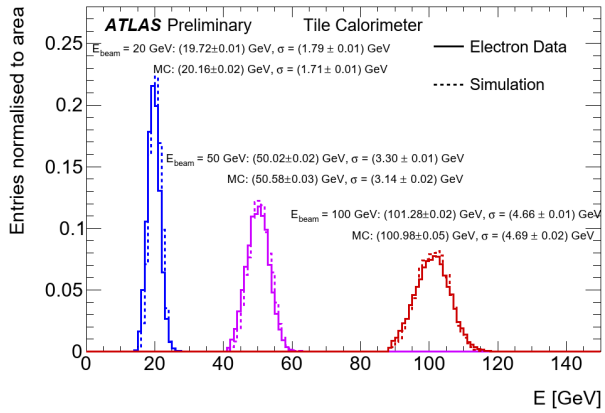


Figure 10: Total energy deposited in the calorimeter obtained using electron beams was also studied [4].

Detector energy response and resolution were studied using the hadron beams with different energies. The results obtained using muons, electrons and hadrons are in agreement with the calibration settings, as shown in Figure 11.

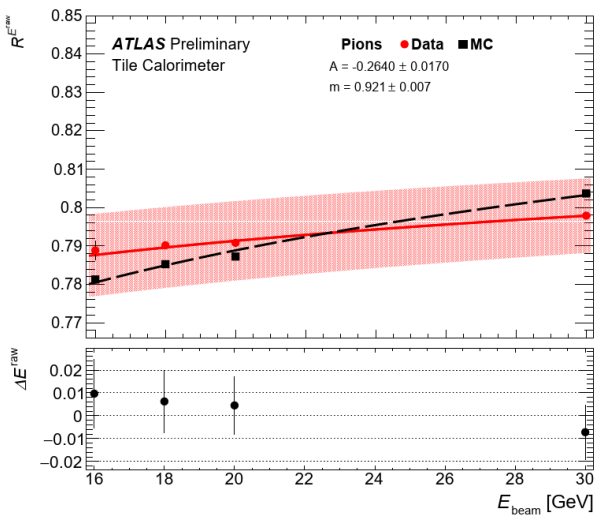


Figure 11: Energy response ratios, measured experimentally (red circles) and predicted by simulation (black squares) as a function of beam energy obtained using pions [4].

Figure 12 shows the stability of the laser over time with low and high gains.

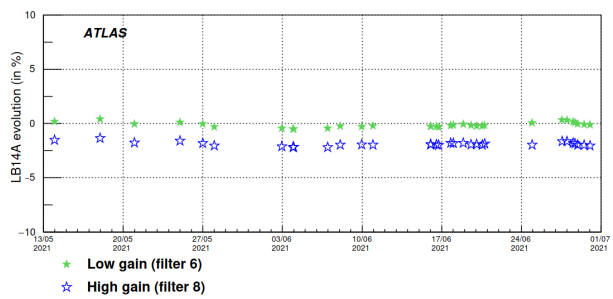


Figure 12: Stability of Laser over time.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV032

Figure 13 and Figure 14 show a comparison of noise between The Demonstrator and Legacy Module.

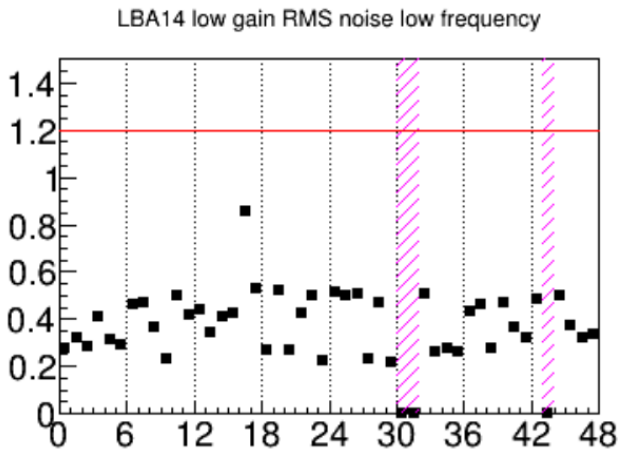


Figure 13: The Demonstrator low gain RMS noise.

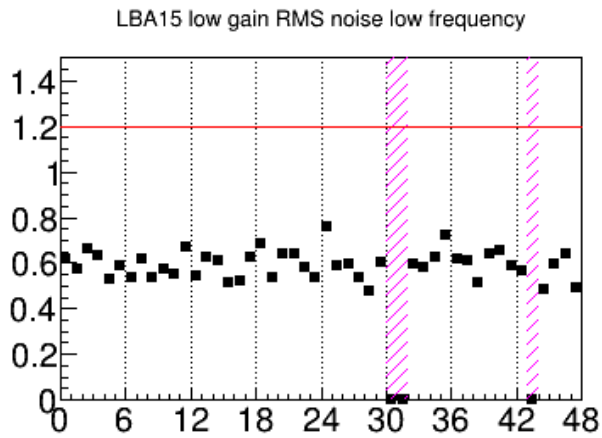


Figure 14: Legacy Module low gain RMS noise.

The above test beam results indicate that The Demonstrator Module performs at least as good as Legacy Module.

SUMMARY

The Tile Demonstrator Module is a prototype for an upgraded readout system and is compatible with current and legacy systems. The Tile Demonstrator is fully integrated into upgraded in the ATLAS Trigger and Data Acquisition and Detector Control systems. It was extensively tested during 2015, 2016 and 2017 test beams and demonstrated good performance. New tests will take place in November 2021 in order to validate new on-detector electronics in the radiation environment and associated off-detector electronics. The Tile Demonstrator module will also be present in Tile Calorimeter during the Run-3 period.

REFERENCE

[1] High-Luminosity Large Hadron Collider (HL-LHC): Technical design report, CERN-2020-010, https://cds.cern.ch/record/2749422

[2] The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 (2008) S08003, https://cds.cern.ch/record/1129811

[3] Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter, ATLAS-TDR-028, https://cds.cern.ch/record/2285583

[4] Upgrade of the ATLAS Hadronic Tile Calorimeter for the High Luminosity LHC, ATL-TILECAL-PROC-2020-009, https://cds.cern.ch/record/2716325

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV033

REUSABLE REAL-TIME SOFTWARE COMPONENTS FOR THE SPS LOW LEVEL RF CONTROL SYSTEM

M. Suminski*, K. Adrianek, B. Bielawski, A. C. Butterworth, J. Egli, G. Hagmann, P. Kuzmanovic, S. Novel Gonzalez, A. Rey, A. Spierer, CERN, Geneva, Switzerland

Abstract

In 2021 the Super Proton Synchrotron has been recommissioned after a complete renovation of its low level RF system (LLRF). The new system has largely moved to digital signal processing, implemented as a set of functional blocks (IP cores) in Field Programmable Gate Arrays (FPGAs) with associated software to control them. Some of these IP cores provide generic functionalities such as timing, function generation and signal acquisition, and are reused in several components, with a potential application in other accelerators.

To take full advantage of the modular approach, IP core flexibility must be complemented by the software stack. In this paper we present steps we have taken to reach this goal from the software point of view, and describe the custom tools and procedures used to implement the various software layers.

INTRODUCTION

The new LLRF system for the SPS has largely replaced old VME-based hardware with a modern installation designed around microTCA crates using PCI-express as the bus.

The update has brought many benefits, one of them being higher hardware density: what used to take a VME crate filled with modules, now is replaced with one or two microTCA cards. It has also affected the control system, as component size has shrunk from a card to an IP core.

Previously, each component was implemented as a VME module, with its own driver, user-space library and application. With the new hardware, the said approach was no longer applicable, therefore a new solution had to be defined.

WORKFLOW

This section will illustrate typical steps needed to develop reusable firmware and software. Overview of the layers constituting a component is presented in Fig. 1.

Memory Map

The process begins with hardware interface definition, starting with individual IP cores and ending with the top level map containing all components used in a card. The interface is called memory map since it defines mapping registers and memories to offsets in a card memory space. Memory map serves as the primary data source for both firmware and software developers.

Each register is described by a number of attributes, such as name, access mode (read-write/read-only), bit width, valid value range or conversion functions between raw register value and physical units.

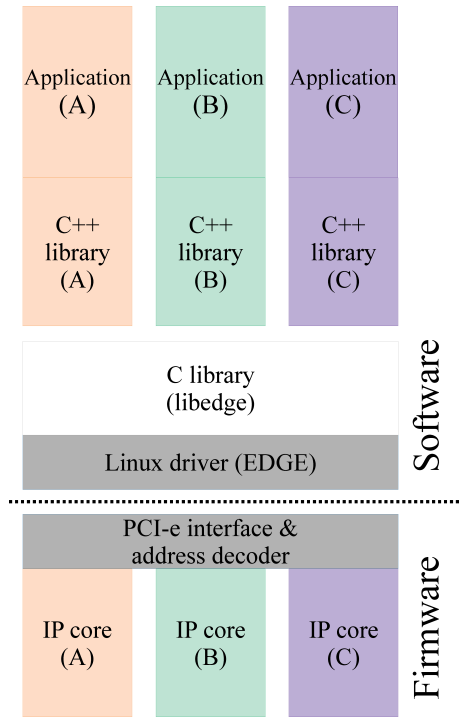


Figure 1: Example of a card implementation with several components. The white block (C library) is common to all cards and components. The grey blocks (Linux driver & address decoder) are card specific. The remaining colors show parts of reusable component stacks, each color representing a different component.

Registers may have subcomponents called fields, providing a way to give different meaning to a set of bits belonging to a particular register. This method is frequently used for status and control registers, where each bit represents a different part of the logic.

Each memory map may also include other memory maps, allowing the designer to reuse existing ones and establish a tree-like structure. The latter method is commonly applied for composing IP cores to define a card interface, also known as the top level map.

Memory maps are edited using a dedicated tool named Reksio (former Cheburashka [1]). The tool offers a graphical user interface aiding the users in memory map creation and validation. It also provides straightforward access to external tools, such as generators for various layers of the component stack.

* maciej.suminski@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Once a memory map is finished, it serves as the source to generate several layers of the component stack:

- FPGA firmware template
- Linux driver
- C++ library

Generated components will be discussed in detail in the following sections.

FPGA Firmware Template

The firmware designer begins with executing the Cheby [2] tool to generate skeleton code in VHDL, creating the foundation for firmware development. The obtained code supplies a bus interface used for data transfers and a set of signals directly representing registers. The latter part is used by the firmware designer to implement the actual component logic.

This step may be executed at the component level, resulting in self-contained, component-specific code or at the top level in order to generate an address decoder for a card.

Linux Driver

Another part of the toolset is the Encore Driver Generator (EDGE) [4], which translates a memory map file to Linux driver source code.

While in certain cases it would be possible to bypass the driver by directly accessing memory-mapped device registers, having a driver brings certain benefits. In a microTCA module, the driver is needed to enable Direct Memory Access (DMA) transfers or handle hardware interrupts.

Apart from driver generation, EDGE also provides a user-space shared C library to communicate with the hardware. The library interface allows the software developer to access any register by its name, using a driver specific memory map file. Thanks to that, the library is common to all components and is not included in the component stack.

It is worth mentioning that EDGE supports both PCI/PCI-express and VME, which is especially convenient when a component needs to be migrated to a different platform.

Please note that the Linux driver is card specific and does not belong to any particular component.

C++ Library

The next layer of the software stack is a C++ library, also generated from a memory map. It uses the EDGE library for accessing the hardware, and as such requires a driver generated with the aforementioned tool. While the C++ library is not strictly required for the application development, it facilitates the software development process.

First of all, generated C++ libraries provide a user-friendly interface, where the tree-like hierarchy of a memory map is reflected in a set of classes, each representing a submap, register or field. Such an approach allows the developer to easily traverse the structure by accessing appropriate fields in the generated classes.

Moreover, the generated library takes advantage of certain properties defined in the memory map:

- Data type (bit width and signedness): ensures correct register value interpretation.
- Valid value range: prevents setting registers to values which are not considered correct.
- Functions to convert between raw register values and physical units: enables writing application code using values expressed in physical units without any additional effort.

Real-Time Application

The last layer of the software stack is the real-time application. At CERN, such applications are developed using the Front-end Software Architecture (FESA [5]) framework with C++ as the programming language.

Most applications built using FESA consist of two parts:

- Server: used for communication with the user or higher level applications. It provides a way to specify new settings or read data acquired from the hardware.
- Real-time: controls the hardware synchronously to the timing system. This part most often configures the card for a particular accelerator cycle and reads back data from the hardware.

Application code might also be generated, but the interface would correspond directly to the memory map. In most cases, applications are expected to implement a user-friendly interface and not one that gives a direct access to the hardware. Usually this layer takes most time to implement.

VERSION VALIDATION

It is usual for components to evolve as new requirements arise. If a change is made to hardware, then it needs to be propagated through each layer upwards in order to avoid unexpected behaviour, which is often difficult to analyze.

For this reason it is essential to verify if all layers of a component are compatible with each other. The simplest way to achieve that is by assigning each layer a version number. In the new SPS LLRF system, the numbers are assigned in accordance with semantic versioning [5] scheme, which defines clear rules regarding which versions are compatible with each other.

For hardware, it has been decided that the first registers in each memory map constitute a standard header providing firmware and memory map version numbers. This is true for both top level and component memory maps.

The Linux driver version is expected to match the firmware version of the card. EDGE offers a way to specify an automatic version check by specifying a register and its expected value range. If the register value is outside of the range, the driver will refuse to load. This step assures memory map compatibility between software and hardware domains.

The user-space EDGE library (libedge) executes another test. When a device is opened, the library reads its driver version and loads the corresponding memory map file, which will be used for obtaining register offsets. At this stage the library also computes the memory map file checksum and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV033

compares it with the one stored in the driver. Further access to the card is possible only when the two numbers match.

Finally, each user-space application is also expected to validate the hardware version. Normally this is done by comparing the component firmware version against the version supported by the application.

EXAMPLES

During the SPS LLRF upgrade, several common components have been developed to be applied in Cavity Controllers and Beam Control microTCA cards:

- Timing generator core: programmable timer generating pulses with requested delays.
- Acquisition core: generic component for sampling data inside the FPGA and transferring it to a memory.
- Resampler core: converts acquired data between fixed sampling rate and beam synchronous rate.
- Function generator core: delivers time series data according to a programmed pattern.
- Numerically controlled oscillator core: generates samples corresponding to a sine wave at the requested frequency, with certain customizations for application in accelerators.
- Gigabit link core: controller for a gigabit serial link used for data transfer.

Each of the above components has an implementation in all the presented layers apart from the Linux driver, which is card specific. If one of them needs to be instantiated in another system, the whole stack can be reused.

CONCLUSION

The described workflow has been successfully applied during the SPS LLRF renovation. The process has delivered several self-contained components, covering all layers from hardware to software application. The components have been reused in two cards without any modification and are potentially applicable in other systems, including non-microTCA ones.

The new approach has improved task coordination in the development process, since several components could be developed independently and tested with any available hardware.

REFERENCES

[1] P. Plutecki, B. Bielawski, and A. C. Butterworth, "Code Generation Tools and Editor for Memory Maps", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 493-496. doi:10.18429/JACoW-ICALEPCS2019-MOPHA115

[2] Cheby, https://gitlab.cern.ch/cohtdrivers/cheby

[3] EDGE Driver Generator, https://gitlab.cern.ch/cohtdrivers/encore

[4] M. Arruat *et al.*, "Front-End Software Architecture", in *Proc. ICALEPCS'07*, Knoxville, Tennessee, USA, paper WOPA04, p. 310-312.

[5] Semantic Versioning, https://semver.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV036

LASER DRIVER STATE ESTIMATION ORIENTED DATA GOVERNANCE

J. Luo†, L. Li, Z. G. Ni, X.W. Zhou, Institute of Computer Application, China Academy of Engineering Physics, Mianyang City, China

Abstract

Laser driver state estimation is an important task during the operation process for the high-power laser facility, by utilizing measured data to analyze experiment results and laser driver performances. It involves complicated data processing jobs, including data extraction, data cleaning, data fusion, data visualization and so on. Data governance aims to improve the efficiency and quality of data analysis for laser driver state estimation, which focuses on 4 aspects — data specification, data cleaning, data exchange, and data integration. The achievements of data governance contribute to not only laser driver state estimation, but also other experimental data analysis applications.

INTRODUCTON

Laser driver state evaluation is an important part of the business process of a laser shooting experiment, which mainly includes physical experiment results evaluation and driver performances evaluation. The energy, pulse power waveform and near-field results of the experimental process are extracted by using the measurement data obtained by acquisition equipments such as energy card meters, oscilloscopes and CCDs in the diagnostic system, and the energy dispersion, power imbalance and other results of a shooting experiment, as well as the gain capacity, output capacity and output stability of the driver are calculated, to evaluate the effectiveness of physical experiments and the performance status of the laser driver.

By adopting the Oracle relational database platform, all the experimental measurement data of the facility are stored [1, 2]. The early database design defined the physical structure and logical structure specification of experimental data storage, but there were no constraints on the storage format of experimental data, so there existed inconsistencies in the storage format of the same type of experimental data. Secondly, there are invalid measurements or missing measurements in the process of experimental data measurement, but such measurement data are not screened when the experimental data are stored in the database, resulting in experimental data quality problems such as missing or abnormal data items. In addition, with the laser facility putting into experimental operations, a large number of experimental measurement data are accumulated, and multiple application systems involving experimental data processing and storage are generated. For these late emerging application systems, most of them do not follow the early database design specifications in terms of data storage. Therefore, the experimental data relationship of the whole facility is complex.

The operational process of a shooting experiment.is described in Figure 1. As an important asset of the facility, the increasingly accumulated experimental data has played an important role in the operation control optimization of the facility in recent years, and the state evaluation of laser driver is one of the main contents. However, limited by the above problems such as inconsistent experimental data storage format, missing experimental data items, abnormal experimental data and complex experimental data relationship, data preprocessing consumes a lot of time, and application systems involving experimental data processing and analysis reveal common problems, such as slow development progress, repeated processing of experimental data, low computational efficiency, and uncertain analysis results, which coincides with the current situation of data preparation mentioned in literature [3].

Data governance aims to improve the efficiency and quality of data analysis and plays an important role in the whole big data analysis process [4]. In recent years, with the rapid development of big data analysis and application research, data governance has played a very significant role in enterprise big data mining, government public data sharing, industrial big data analysis, scientific research data management and other industries [5-12]. The state estimation of a laser driver involves a lot of jobs, like experimental data extraction, transformation, joint calculation and visualization. Considering the current situation of the experimental data quality and the problems faced by the data processing application system, we utilize a framework consisting with data governance and data visualization to implement the laser driver state estimation system. This paper mainly introduces and summarizes our work about the data governance part.

The next section provides an overview of laser driver state estimation. Section III illustrates the data governance. The computation of state parameters of laser driver is discussed in section IV. Finally, we conclude this paper in section V.

LASER DRIVER STATE EVALUATION

The main purpose of state estimation of laser driver is to evaluate the compliance of physical experiment and the performance of the laser driver. The energy, pulse power waveform and near-field results of the experimental process are extracted from the original measurements. And the key technical parameters such as power imbalance, output beam quality, output stability, gain capability and third harmonic efficiency of the laser driver are calculated. Furthermore, their evolution law is visually analyzed.

† luoj1987@caep.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV036

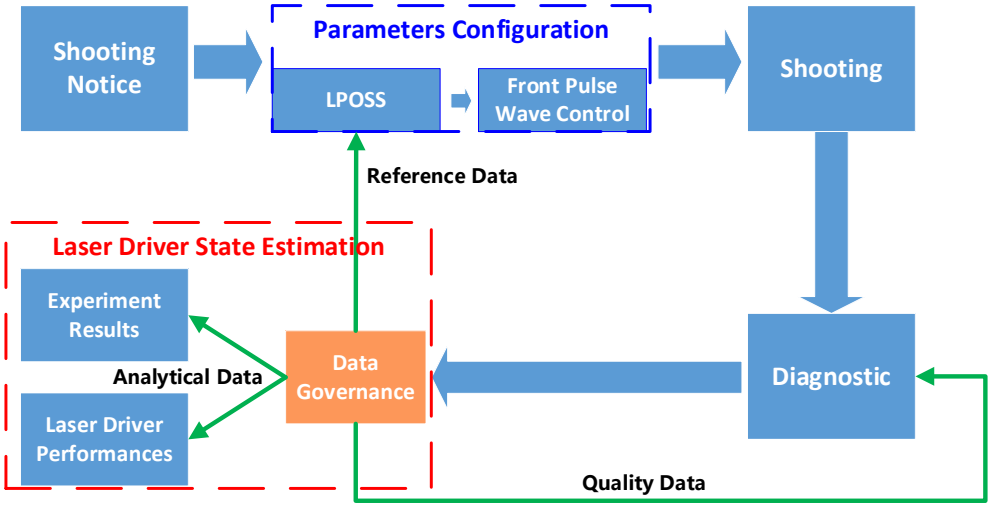


Figure 1: Business process of a shooting experiment.

Evaluation Contents

The state evaluation of the laser driver mainly includes two aspects: the evaluation of physical experiment results and the evaluation of laser driver performance.

a. State of physical experiment results: including output energy and pulse power waveform of each laser beam, total output energy, energy dispersion of each beam set, total average power and power imbalance. Among them, energy dispersion and power imbalance are two important criterion to judge whether the physical experiment meets the target or not, and they are also key factors to indicate the energy and power balance of the laser driver.

b. State of laser driver performance: including driver gain capability, output capability and output stability. Among them, the gain capability includes the main amplification gain and pre amplification gain, the output capability mainly refers to the peak power of each laser beam, and the output stability includes the frequency conversion efficiency and the energy deviation of each laser beam.

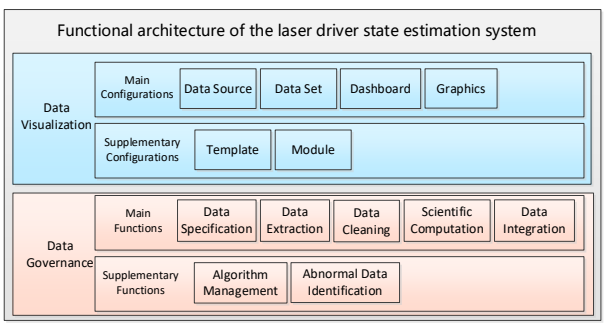


Figure 2: Functional architecture of the laser driver state estimation system.

Functional Architecture

The laser driver state estimation system mainly consists with two functional parts, namely data governance and data visualization, as shown in Figure 2.

Data visualization is mainly responsible for the interface display of driver state parameters, while data governance realizes the whole process of data extraction, cleaning, calculation, transformation and storage integration, as well as the functions of analysis and processing algorithm management and anomaly detection.

Technical Architecture

The laser driver state estimation system is composed of a hierarchical architecture, including data application layer, data management layer, data computation layer and data sources layer. The technical architecture is shown in Figure 3. Data governance mainly covers the data computing layer and the data management layer.

a. Data sources: in the process of experimental operation, the original experimental data is generated and stored in Oracle database. The data forms are diversified, including numerical value, text, picture and other data.

b. Data computation: extracting the original measurement data from the data source system, then cleaning the source data, detecting the abnormality and mark the quality of the measured data. Finally, combined with the statistical analysis algorithm, the state evaluation result data of the laser driver are obtained.

c. Data management: according to certain physical structure, logical structure and data format specifications, the analyzed state parameters are stored in the data warehouse for centralized management. In the data management layer, data is stored in Oracle relational database.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV036

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

d. Data application: providing the data in the data warehouse system to the end user in the form of chart through the web interface. According to the analysis needs of users, report, query, multi-dimensional analysis and other functions are used to display data.

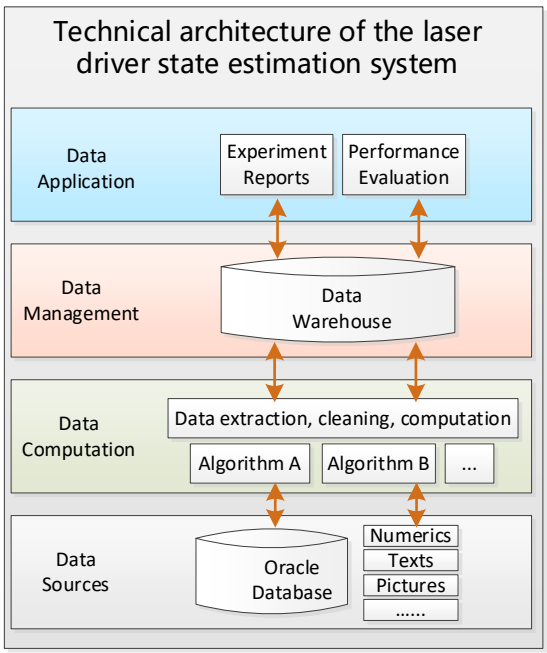


Figure 3: Technical architecture of the laser driver state estimation system.

DATA GOVERNANCE FOR LASER DRIVER STATE EVALUATION

The data governance is the process of reviewing and transforming data in order to make data better serving data analysis. It is the most energy consuming part in the whole data analysis process [3]. Although the original motivation of conducting the data governance is to conveniently serve the laser driver state evaluation system, once the data governance is carried out, it becomes a task of dynamically constructing data processing process for various business systems. Data governance pre-processes data for specific problems, adopts different data processing algorithms for different problems, and some data governance achievements can be shared among multiple applications. The final result of data governance for laser driver state estimation can serve all relevant data users in the whole facility, as shown in Figure 1.

Data governance mainly includes four aspects: data specification, data cleaning, data exchange and data integration. The work in these four aspects is described in detail below.

Data Specification

Aiming at the problem that the data formats of the same type of data are inconsistent, the data formats of pulse power waveform data and CCD image data are designed uniformly.

Pulse power waveform data is represented by [time column, power column] two-dimensional double array. The background measurement results outside the effective time window in the original measurement waveform are no longer retained, and only the part within the effective time window in the original measurement waveform is selected, as depicted in Figure 4.

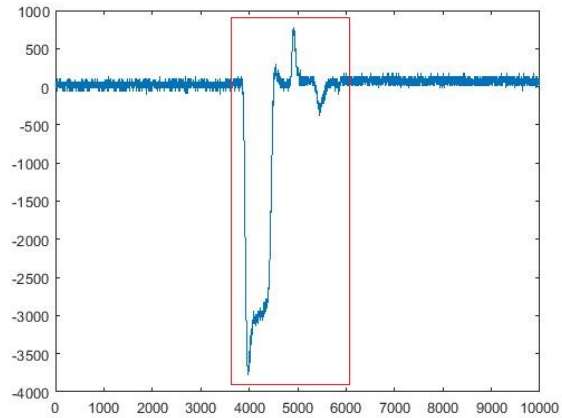


Figure 4: Example of pulse waveform data (red rectangle indicates the effective time window).

CCD image data is represented according to integer pixel matrix. Each element in the matrix represents the pixel gray-scale value at the corresponding position of the image. In order to facilitate subsequent data analysis, CCD image data is divided into two types, original measured and intended clipped. The matrix dimensions of these two types are different. The original measurement is fixed at 1024 * 1024 scale, and the pixel matrix dimension of the clipped image depends on the effective illumination range during actual imaging, as shown in Figure 5.

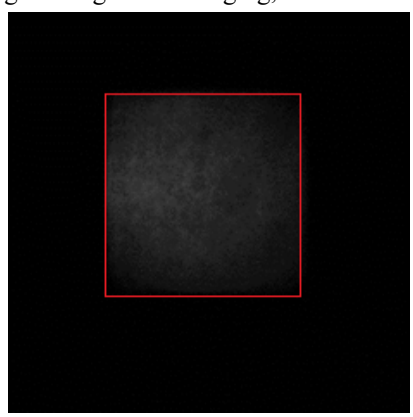


Figure 5: Example of CCD image data (red rectangle indicates effective illumination range).

Data Cleaning

Data cleaning is a necessary step for most data-driven tasks. Data cleaning refers to detecting and correcting possible errors from data to ensure the quality of data and

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV036

comply with domain related integrity constraints. The data cleaning mainly refers to the integrity and anomaly detection of the original measurement data. Integrity detection is to judge whether the laser beam participating in the experiment has energy, waveform and near-field measurement results at all positions based on the experimental configuration parameters. Anomaly detection is to judge whether there is a serious mismatch between the measurement results and the actual output, which will significantly affect the state evaluation results.

Data Exchange

A data governance service named "report and performance calculation process" is established to exchange data through the database read-write interface.

The service reads the experimental measurement data from the original database of the data source layer through the database interface, realizes the functions of data extraction, data cleaning, analysis and calculation in combination with the algorithm, and stores the analytical results in the data warehouse of the data management layer, for the visualization function module to directly read the data and display the charts.

The "report and performance calculation process" service is integrated into the main operation workflow. It is called by the centralized control system. After each shooting experiment, the experimental data governance of the shot is started to calculate the state parameters of the laser driver.

Data Integration

Data integration is the physical or logical integration of heterogeneous, distributed and autonomous data to provide users with a unified access interface, so that they can access data sources transparently, so as to realize comprehensive data sharing [3,4]. There are two common schemes for data integration, one is pattern integration method, and the other is data replication method, such as data warehouse. This paper utilizes the data replication method, based on the logical model of experimental number, laser beam number, experimental configuration parameters, experimental measurement results and state evaluation results, constructs the data warehouse of driver state evaluation. The visualization function module directly reads data from the data warehouse and displays a variety of charts.

CALCULATION OF LASER DRIVER STATE PARAMETERS

The contents of laser driver state estimation are described in Section 2.1, mainly including physical experiment results and laser driver performances. In order to calculate these data, we need to focus on two important issues: the data flow and the calculation algorithm.

Data Flow

The relationship between experimental configuration parameters, original measurement data and state parameters data is shown in Figure 6.

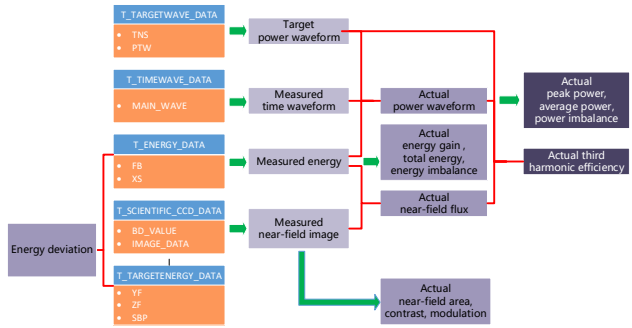


Figure 6: Data flow of calculating state parameters.

Agorithm Integration

The data governance service for the laser driver state estimation follows the software architecture and integration specification of the centralized control system, and is coded in C++. Data governance service involves various data processing algorithms in data extraction, data cleaning, analysis and calculation. By weighing the difficulty of algorithm implementation and the time efficiency of calculation, the matlab is mostly used. These algorithms are encapsulated into dynamic link library files for data governance services to call. According to the data relationship as shown in Figure 6, the data input and output interfaces of each algorithm are almost fixed. In case of algorithm upgrade, we only need to replace the corresponding algorithm dynamic link library file and rebuild the data governance service.

CONCLUSION

This paper summarizes the data governance work for the application of laser driver state estimation from four aspects: data specification, data cleaning, data exchange and data integration. The final result can serve all relevant data users in the whole laser facility.

Data governance involves various data processing algorithms. As a supplement to traditional algorithms, artificial intelligence algorithms increasingly show a big potential. Our future research will be carried out in the development and integration of intelligent algorithms to gradually optimize the effect of data governance.

REFERENCES

[1] Xiaowei Zhou, Daojian Yao, *et al.*, “Centralized control software system of XX unit - database design report”.

[2] Daojian Yao, “Design of computer centralized control system for large laser device”, National Defense Technology (GF) report database of Chinese Academy of physics, 2013.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV036

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[3] Xiaoyong Du, Yueguo Chen, Ju Fan, Wei Lu "Data collation - Key Technology of big data governance [J]", Big Data, 2019, 3:13-22.

[4] Xindong Wu, Bingbing Dong, Xinzheng Du, Wei Yang, "Data governance technology [J]", Journal of Software, 2019, 30 (9): 2830-2856.

[5] Jian Yin, Huaijie Zhu, Jianxing Yu, Shuang Qiu, "Panoramic framework of big data governance [J]", Big Data, 2020, 2:19-26.

[6] Xiaoou Ding, Hongzhi Wang, Shengjian Yu, "Quality management of industrial time series big data [J]", Big Bata, 2019, 6:19-29.

[7] Yike Guo, Wei Pan, Simiao Yu, Chao Wu, Shicai Wang, "Big Data for Science [J]", Journal of the Chinese Academy of Sciences, 2016, 6 (2): 599-607.

[8] Yaping Yang, Yi Wang, Yan Bai, Xiafang Le, Jia Du, Yongqing Bai, Jiulin Sun, "Development and practice of national earth system science data center [J]", Journal of Agricultural Big Data, 2019, 1 (4): 5-13.

[9] Senlin Xiong, Ziming Zou, Xiaoyan Hu, Zhen Ji, "Spatial science data product organization model [J]", Journal of Agricultural Big Data, 2019, 1 (4): 30-36.

[10] Guomin Zhou, Jingchao Fan, "Design and implementation of agricultural science observation data aggregation management platform [J]", Journal of fan, 2019, 1 (3): 38-45.

[11] Guoliang Li, JiannanWang, Yudian Zheng, Franklin M. J., "Crowdsourced data management: a survey[J]", IEEE Transactions on Knowledge and Data Engineering, 2016, 28(9): 2296-2319.

[12] Daniel E. O'Leary, "Embedding AI and Crowdsourcing in the Big Data Lake[J]", IEEE Intelligent Systems, 2014, 29(5): 70-73.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV037

THE IMPLEMENTATION OF THE BEAM PROFILE APPLICATION FOR KOMAC BEAM EMITTANCE*

Jae-Ha Kim†, Young-Gi Song, SungYun Cho, Seunghyun Lee, Sang-Pil Yun
Korea Multi-purpose Accelerator Complex, Korea Atomic Energy Research Institute, Gyeongju, Korea

Abstract

Korea Multi-purpose Accelerator Complex (KOMAC) has been operating a 100 MeV proton linear accelerator that accelerates a beam using ion source, a radio frequency quadrupole (RFQ), 11 drift tube linac (DTL). And the accelerated protons are transported to target rooms that meets the conditions required by the users. It is important to figure out the beam profile of the proton linac to provide the proper beam condition to users. We installed 8 wire scanners to measure beam emittance of KOMAC at beam lines. And beam profile application to measure beam emittance has been implemented using EPICS and python. This paper will describe the implementation of the beam profile application for KOMAC beam emittance.

INTRODUCTION

KOMAC has been operating five beamlines and five target rooms for clients. A proton beam that clients require is transported to a target room through a beamline. So it is important to identify the characteristics for the proton linac and a proton beam to provide and appropriate beam. Therefor KOMAC installed Beam profile measurement devices that are Beam Position Monitor, Beam Loss Monitor, Beam Phase Monitor to measure beam characteristics and eight wire scanners at beamlines that are TR23, TR103, TR104, TR105 and straight beamline to figure out the beam profile of the proton beam. Following Fig. 1 shows layout of KOMAC linac and beamlines.

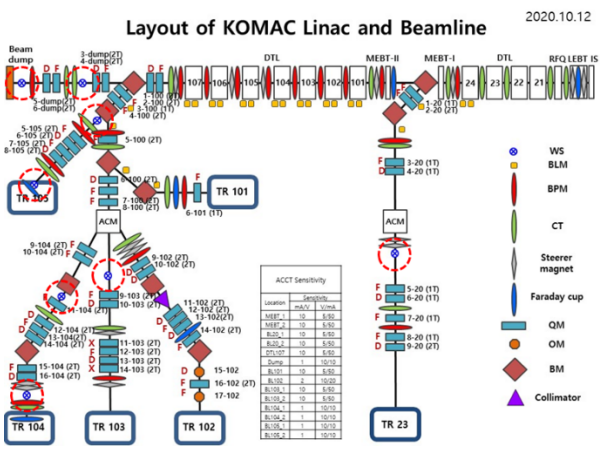


Figure 1: the layout of KOMAC linac and beamlines.

KOMAC control system based on Experimental Physics and Industrial Control System (EPICS) framework has been implemented to control the 100 MeV linac and peripheral devices at the control room [1]. Figure 2 shows the block diagram of KOMAC control system.

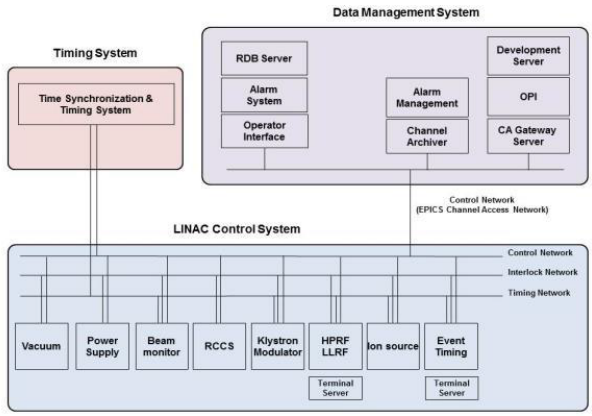


Figure 2: Block diagram of KOMAC control system.

We adopted Control System Studio (CSS) for user interface and Archiver Appliance for data saving system [2], [3]. And we have been developed high level applications that can communicate with the control system to study the beam profile of the linac. To integrate the wire scanner into KOMAC control system, the wire scanner control system was implemented using EPICS to control wire scanner and pyEPICS for data process [4].

WIRE SCANNER

The wire scanner has been developed to measure beam emittance of the KOMAC linac. The specification of the wire scanner is shown in Table 1.

Table 1: The Specification of The Wire Scanner

Specification	
Wire material	W tungsten
Wire diameter	0.1 mm
Moving speed & Range	100 mm/s, 50 mm (± 25 mm)
Spatial accuracy	0.05 mm
Spatial resolution	0.1 mm
Mounting Flange	6” CF

The wire scanner control system is divided into two part: driving wire scanner; data analysis. The layout of the existing wire scanner control system is shown in Fig. 3.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV037

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

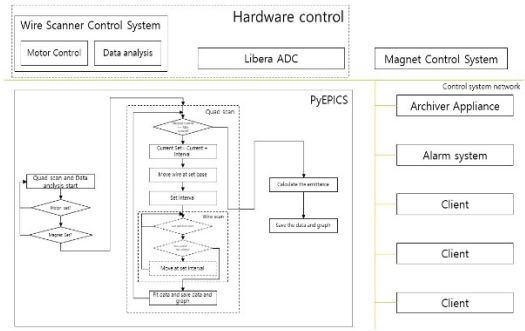


Figure 3: The existing wire scanner control system.

The part of driving wire scanner controls a position of the wires using the motor and gets the current data from the wires using libera ADC. The data process part analyses the obtained data from wire scanner.

Wire Scanner Control System

EPICS IOC controls a motor to move a position of a wire in wire scanner by communicating with a motor controller. A Libera digit electronic captures data whenever a beam passes the wires and delivers the data to EPICS IOC. For the convenience of managing the wire scanners, a wire scanner control unit that is made up of two motor controllers and a libera digit electronic, is created and installed as shown Fig. 4 and Fig. 5.

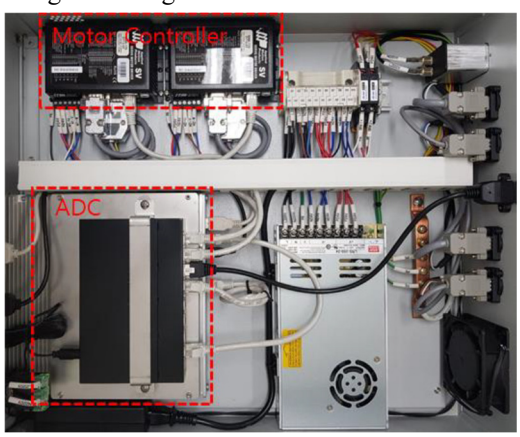


Figure 4: The wire scanner control units.

Figure 5: The wire scanner control units installed at KOMAC linac.

Quad Scan Interface with pyEPICS

pyEPICS that can access the EPICS IOC using Channel Access protocol was adopted for the quad scan interface. the quad scan interface using pyEPICS based on Python programming language takes data from EPICS IOC in synchronization with the beam repetition rate. The algorithm diagram of quad scan and data analysis is shown in Fig. 6.

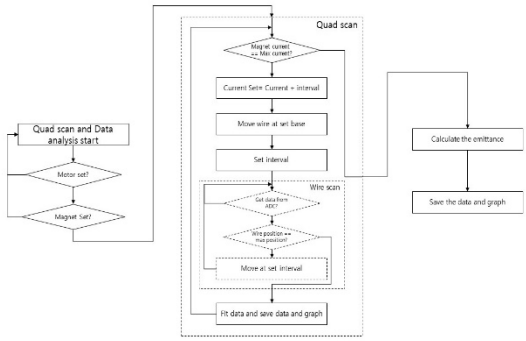


Figure 6: The algorithm diagram of the quad scan interface.

After quad scanning, the quad scan interface delivers the processed data to the EPICS IOC, and all the data are archived by the Archiver appliance that is chosen for the KOMAC data archiving system. In addition, the processed data are saved in file and picture format. However, quad scan interface must be running in the background as a daemon. And when a modification occurs, we need to modify both EPICS IOC and python code

Quad Scan Interface with sscan and asub Record

To simplify the process of the quad scan and wire scan, All the process has been integrated into the EPICS IOC. So for implementation of the quad scan interface, sscan EPICS modules and asub record were adopted [5], [6]. The sscan module performs the quad scan algorithm previously implemented by pyEPICS. The EPICS database for quad scan is shown in Fig. 7.

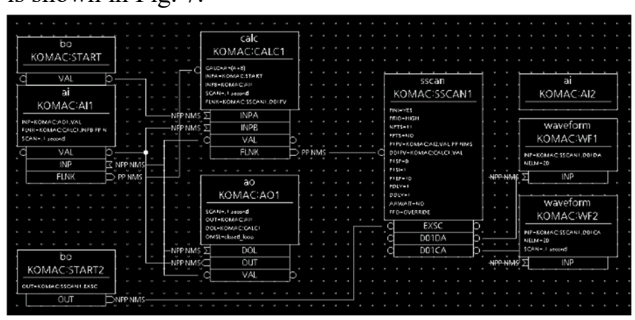


Figure 7: The block diagram of the EPICS database for the quad scan.

Quad scanning is started with the start button and the sscan module moves the wire and acquire data of a proton beam according to a beam repetition until the wire is moved at the set position. The obtained data are passed to the waveform record and are shown on CSS in real-time.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV037

After scanning, all the data are delivered to asub record and are analyzed. Figure 8 shows the raw data and the fitted data after quad scanning.

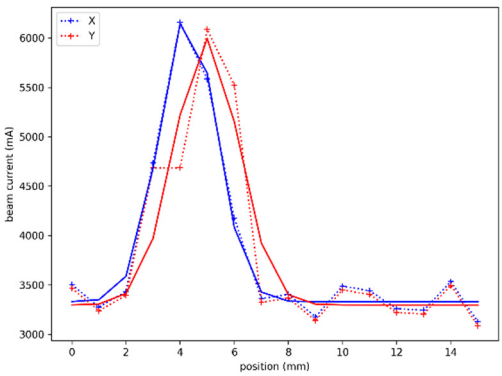


Figure 8: The raw data da fitted data from quad scan.

The obtained data are saved in text and graph file format. And asub record reads all the text file resulted from quad scan and calculated the rms beam size versus Field gradient that is shown in Fig. 9.

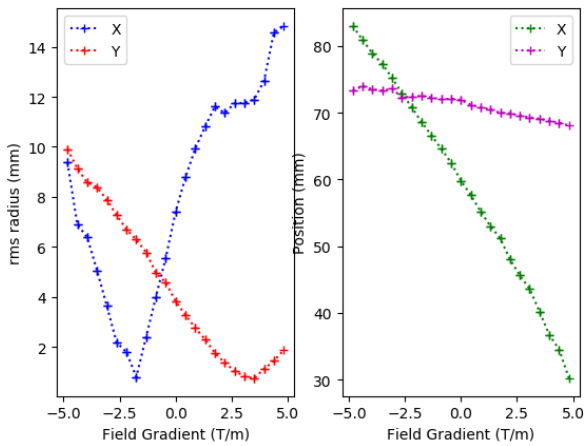


Figure 9: The rms beam size versus Field gradient.

The beam emittance of the 100 MeV proton linac is then obtained from the rm beam size versus field gradient. Following Fig. 10 shows the beam emittance in test environment.

.

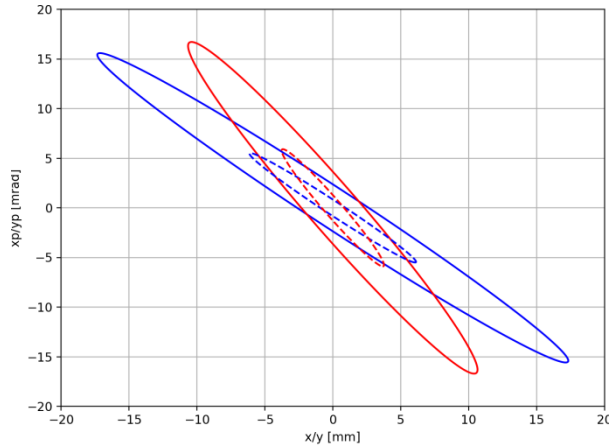


Figure 10: The beam emittance of the linac in test environment.

CONCLUSION

The beam profile application for the KOMAC beam emittance has been integrated into KOMAC control system. The wire scanner control system performs the quad scan and data processing automatically. By using sscan and asub record, the process of the quad scan has been reduced. The program shows the wire scan data in real time and then calculated the fitted data and beam emittance after quad scanning. In the future, the beam profile application is applied to KOMAC 100 MeV proton linac to conduct the test.

ACKNOWLEDGEMENTS

This work has been supported through KOMAC (Korea Multi-purpose Accelerator Complex) operation fund of KAERI by MSIT (Ministry of Sci-ence and ICT).

REFERENCE

[1] Experimental Physics and Industrial Control System (EPICS), http://www.aps.anl.gov/epics

[2] Control System Studio (CSS), https://controlsystem-studio.org

[3] The EPICS Archive Appliance, http://slacmshankar.github.io/epicsarchiver_docs

[4] PyEpics, https://pyepics.github.io/pyepics/index.html

[5] sscan – EPICS, https://epics.anl.gov/bcda/synApps/sscan/sscan.html

[6] Array Subroutine Record (aSub) – EPICS, https://epics.anl.gov/base/R7-0/4-docs/aSubRecord.html

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV038

PLUG-IN-BASED PTYCHOGRAPHY & CDI RECONSTRUCTION USER INTERFACE DEVELOPMENT

Sang-Woo Kim*, Kyung-Hyun Ku, Woul-Woo Lee
Pohang Accelerator Laboratory, Pohang, South Korea

Abstract

Synchrotron beamlines have a wide range of fields, and accordingly, various open source and commercial softwares are being used for data analysis. Inevitable, the user interface differs between programs and there is little shared part, so the user had to spend a lot of effort to perform a new experimental analysis and learn how to use the program newly.

In order to overcome these shortcomings, the same user interface was maintained using the Xi-cam framework [1], and different analysis algorithms for each field were introduced in a plugin method. In this presentation, user interfaces designed for ptychography and cdi reconstruction will be introduced.

INTRODUCTION

With the development of technology, the amount of data from detectors used for synchrotron radiation facilities is rapidly increasing. Diverse programs for processing the massive data are being developed indivisually at synchrotrons and universities with various languages such as python, matlab, labview, etc. In this situation, Xi-cam [1] proposed to jointly develop data analysis software with standardized framework and to utilize the developed program in other research institutes. Analysis programs using standardized framework have the advantage of being able to easily add necessary functions by loading plugins while maintaining a unified user interface. In addition, since the Xi-cam program handles GUI and analysis code execution, it is possible to focus on the implementation of core functionality. For this reason, we created a ptychography and cdi data processing program using Xi-cam's framework.

Ptychography and CDI are two types of coherent X-ray imaging techniques. As the focused and coherent X-rays pass through the sample, an interference pattern is created and this pattern is recorded by the detector. By using the interference pattern, the wavefunction of the probe, the density distribution of the sample, and the strain information could be obtained. The method estimates the wave function of the probe and the distribution of the sample at the beginning, and repeats the process of updating the probe and sample information to match the measured inteference pattern while repeating propagaion and backward propagation to obtaion a value close to the actual probe and sample. Propagation is mathematicall equivalent to a fourier transformation. When CPU is used to repeatedly perform FFT and inverse FFT of large image data, it takes a long time to obtain a result, whereas GPU could significantly reduce operation time because multiple cores can operate in parallel.

Ptychography and CDI operations were performed using the PyNX [2] developed by ESRF. PyNX was chosen because it is designed to increase the operation speed by using cuda cores of GPU and is designed based on mathematical operator so that the user can modifiy the application of the algorithms to suit their data. Although PyNX provides a script that can apply the reconstruction algorithm from the command line, a graphical user interface was created so that general users who are not familiar with command-line can utilize it.

PTYCHOGRAPHY

Figure 1 shows the intial screen of Xi-cam. A list of installed plugins is displayed in the upper right corner. Currently, ptychography and cdi plugins are displayed, and plugins can be added on demand. Figure 2 is the screen displayed when the Ptychography plugin is selected. Tabs are arranged in the order of data processing, Preprocessing, STXM Viewer, Reconstruction, and Result Viewer. Similarly, CDI plugin (Fig. 6) consists of Preprocessing, Reconstruction, and Result Viewer.

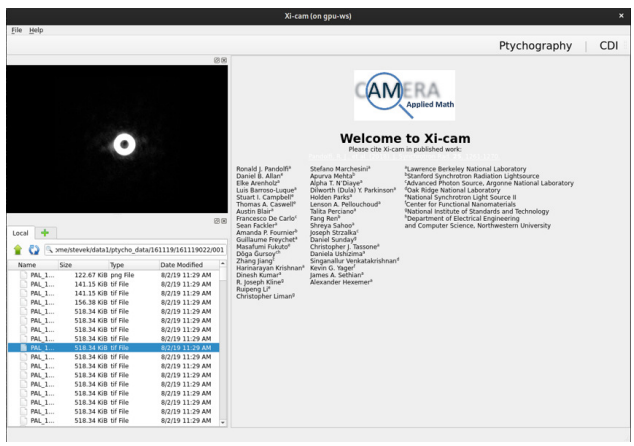


Figure 1: The initial screen of the Xi-cam. Installed plugins can be selected from the upper right corner.

Preprocessing

In preprocessing, meta data such as energy and the distance between the sample and the detector is read from the header file and average, background, substraction, crop, resize, denoising, etc. are performed and then saved as a cxi file. The cxi file format [3] is used as a standard in the field of coherent X-ray imaging and is supported by many programs. Thousands of images are compressed and stored in a single

* physwkim@postech.ac.kr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV038

file, so the efficiency of management and data transmission is increased.

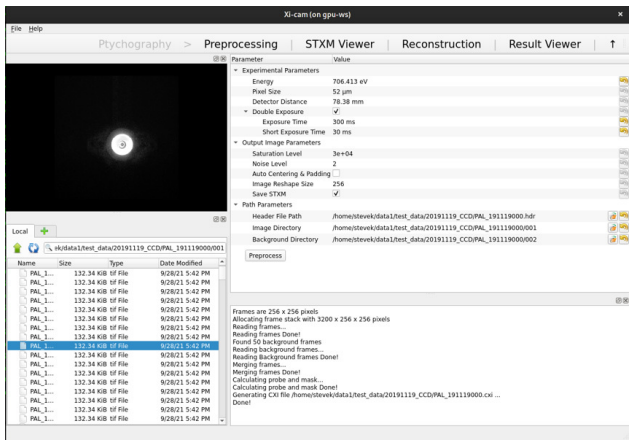


Figure 2: The first screen displayed when the Ptychography plugin is selected. The tabs are arranged in the order of Preprocessing, STXM Viewer, Reconstruction, and Result Viewer workflow.

Reconstruction

In the Reconstruction tab (Fig. 3), the initial probe size and shape are determined, and reconstruction is performed by combining algorithms to be applied. Reconstruction may not work well depending on the direction in which the camera is mounted. In this case, it is possible to find the condition for good reconstruction when the image is transposed or flipped. On the right side, there is a window that updates the amplitude and phase of the sample and the amplitude and phase of the probe while the calculation is in progress. The black square box displayed on the object indicates the area where the actual scan was made. In the lower right log window, the name of the currently running algorithm, the error value, the elapsed time, and the size of the probe finally obtained are displayed.

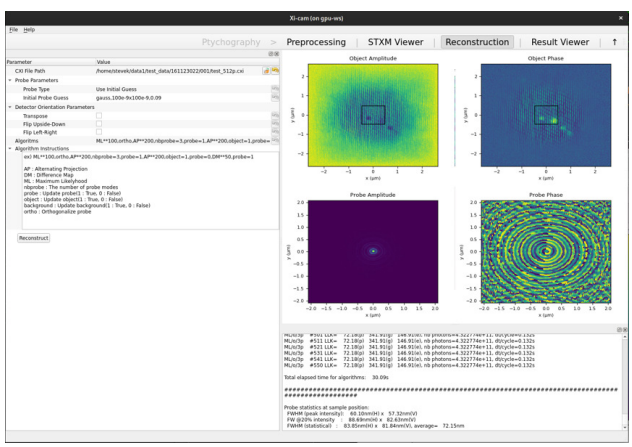


Figure 3: Ptychography reconstruction panel is shown. In the left setting window, the initial probe size, and reconstruction algorithm could be changed.

Since the wave function of the probe can be obtained through the ptychography, it has the advantage of accurately grasping the size and shape of the beam. In addition, if this data is applied to the next operation, a good reconstruction result could be obtained with much fewer operations.

Result Viewer

Figure 4 shows the Result Viewer tab. In this tab, the reconstruction results can be immediately checked and exported or captured. The left panel shows the hierachy of a cxi file. Not only the reconstruction result, but also meta data such as pixel size and energy and the wave function of the probe could be accessed. The right panel shows the amplitude of the reconstructed sample. The black square box indicates the area where the actual scan was made.

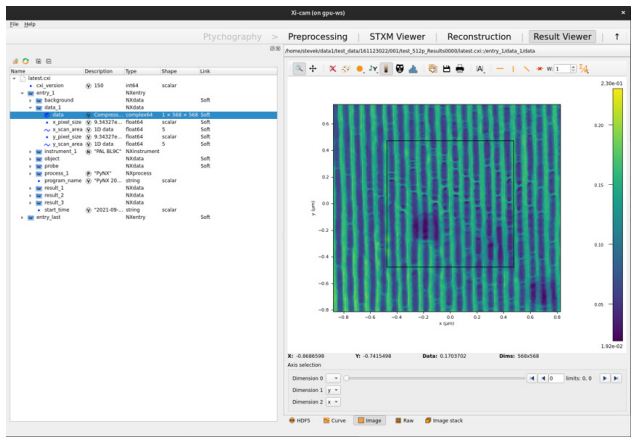


Figure 4: The Result Viewer tab of the Ptychography Plugin. The hierachy of the CXI file resulting from the reconstruction is shown on the left, and the operation result image is shown on the right. The black box indicates the scan area.

Figure 5 shows the STXM (a) and Ptychography reconstruction results (b) of measuring a zoneplate with defects. STXM data were measured in 50 nm steps when the beam size was 60 nm (FWHM). In the reconstruction result, the pixel size was reduced from 50 nm to 9.3 nm, and the noise seen in the STXM data was also greatly reduced in the result. Since the image quality can be improved through the reconstruction, it has many advantages when performing 2D XANES and Tomography experiments using the STXM technique. The soft X-ray beamline 10A1 and the hard X-ray beamline 9C are using the Ptychography plugin.

CDI

Unlike the case of ptychography, the wavefunction of the probe cannot be obtained through CDI reconstruction. Instead, designate an arbitary support area that is slightly larger than the actual sample and perform the reconstruction until it approaches the actual sample while narrowing or increasing the support area. Through this support update, when the boundary of the actual sample and that of the support match, the error is minimized and the information of the sample could be obtained.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV038

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

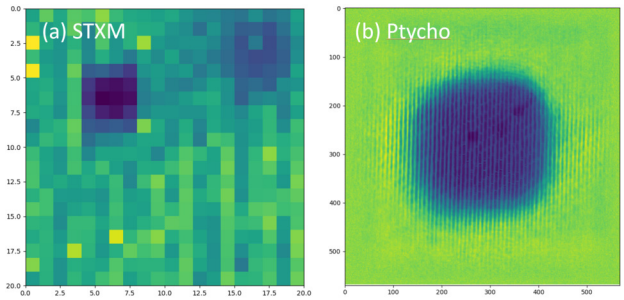


Figure 5: The result of measuring the zoneplate at PLS-II 10A1 beamline is shown. The left (a) is the data measured by typical STXM, and the right (b) is the result obtained after performing the Ptychography reconstruction. The pixel size of (a) is 50 nm, and the pixel size of (b) is 9.3 nm.

The Bragg CDI technique can obtain the 3 dimensional characteristics of the sample in real space by using the interference pattern obtained from the rocking curves around the diffraction peak. The images obtained through rocking measurement is converted into q-space during preprocessing to obtain diffraction patterns in reciprocal space.

Reconstruction

Figure 6 shows the reconstruction tab of the CDI plugin. The parameters of support type and size and reconstruction algorithm setting screen are located on the left. On the right screen, the amplitude and phase of the sample, the reconstructed inteference pattern and the measured interference pattern are shown. The sample distribution in real space was revealed through CDI reconstruction of inteference patterns in 3D q-space.

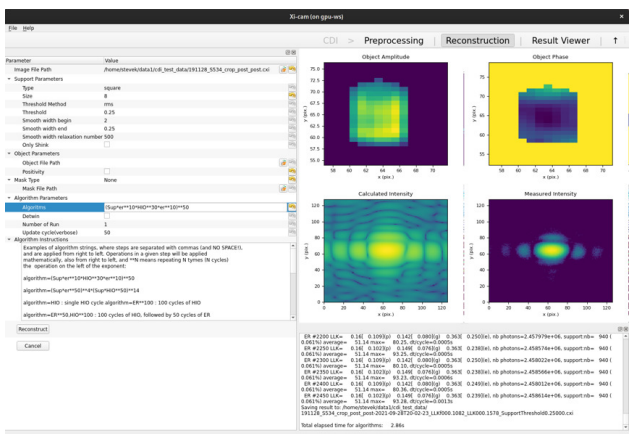


Figure 6: The reconstruction tab of the CDI plugin. Setting parameters and status results are displayed.

Result Viewer

Figure 7 shows the result viewer tab where the CDI reconstruction result can be viewed as a 3D image. The 3D image was displayed using the image viewer of silx [4], and the cross-sectional image of the smaple can be displayed.

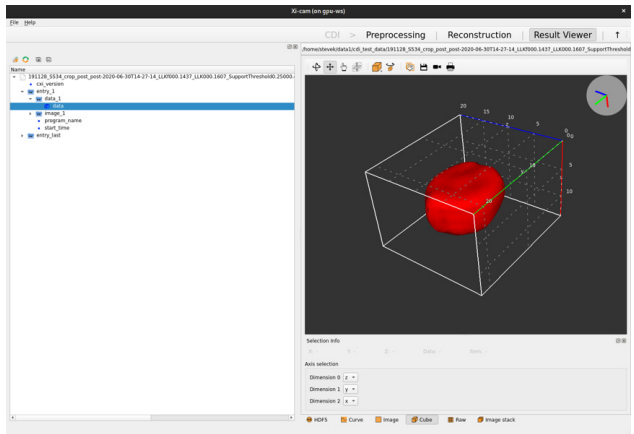


Figure 7: Result Viewer tab of CDI plugin. A three-dimensional image of the reconstructed sample is displayed.

CONCLUSION

Using Xi-cam as a standard framework for data analysis, Ptychography [5] and CDI [6] plugins were made. The unitiy of the user interface has been increased, and the efficiency has also been increased as it is possible to focus only on the production of core functionalities.

The Ptychography and Bragg CDI plugins were applied to the hard X-ray and soft X-ray beamlines. It is comfirmed that image quality can be improved with ptychography compared to stxm and 3D sample information could be obtained through Bragg CDI.

REFERENCES

[1] R. J. Pandolfi *et al.*, "Xi-cam: a versatile interface for data visualization and analysis", *J. Synchrotron Radiat.*, vol. 25, Part 4, p. 1261, Jul. 2018.

[2] V. Favre-Nicolin *et al.*, "PyNX: high-performance computing toolkit for coherent X-ray imaging based on operators", *J. Appl. Cryst.*, vol. 53, p. 1404, Aug. 2020.

[3] Maia, F. R. N. C., "The Coherent X-ray Imaging Data Bank", *Nat. Methods*, vol. 9, pp. 854–855, 2012.

[4] Thomas VINCENT *et al.*, "silx-kit/silx: v0.11.0: 03/07/2019", *Zenodo*, 03-Jul-2019.

[5] `https://bitbucket.org/physwkim/ptychography/src/pls/`.

[6] `https://bitbucket.org/physwkim/xicam.cdi/src/master/`.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

NEW MACHINE LEARNING MODEL APPLICATION FOR THE AUTOMATIC LHC COLLIMATOR BEAM-BASED ALIGNMENT

G. Azzopardi*,1, G. Ricci[1,2]
[1] CERN, Geneva, Switzerland,
[2] Sapienza Università di Roma

Abstract

A collimation system is installed in the Large Hadron Collider (LHC) to protect its sensitive equipment from unavoidable beam losses. An alignment procedure determines the settings of each collimator, by moving the collimator jaws towards the beam until a characteristic loss pattern, consisting of a sharp rise followed by a slow decay, is observed in downstream beam loss monitors. This indicates that the collimator jaw intercepted the reference beam halo and is thus aligned to the beam. The latest alignment software introduced in 2018 relies on supervised machine learning (ML) to detect such spike patterns in real-time. This enables the automatic alignment of the collimators, with a significant reduction in the alignment time. This paper analyses the first-use performance of this new software focusing on solutions to the identified bottleneck caused by waiting a fixed duration of time when detecting spikes. It is proposed to replace the supervised ML model with a Long-Short Term Memory model able to detect spikes in time windows of varying lengths, waiting for a variable duration of time determined by the spike itself. This will allow for further speeding up the automatic alignment.

INTRODUCTION

The CERN Large Hadron Collider (LHC) is the largest particle accelerator in the world, built to accelerate and collide two counter-rotating beams towards the unprecedented design center-of-mass energy of 14 TeV [1]. The LHC is susceptible to beam losses which can damage the state of superconductivity of its magnets [2]. A multi-stage collimation system, consisting of 123 collimators [3], is installed in the LHC. Each collimator consists of two parallel absorbing blocks, referred to as jaws, inside a vacuum tank. The collimators must be aligned with the beam by symmetrically positioning the jaws on either side. This provides a 99.998 % cleaning efficiency of halo particles, preventing any LHC damage [4]. Each year of LHC operation begins with a commissioning phase which involves aligning all collimators and ensuring the correct settings for nominal operation [5].

This paper presents an analysis of first-use performance of the latest alignment software, that makes use of machine learning. A comparison with the previous alignment software resulted in identifying a bottleneck that restricts alignment efficiency. This is followed by a detailed analysis of a Long-Short Term Memory model that can be introduced to further improve the performance of the new software.

* gabriella.azzopardi@cern.ch

BACKGROUND

Recap. of Collimator Alignments

Collimation alignment at the LHC is essential for beam performance and is based on different beam-based techniques developed for the specific LHC conditions [6]. While the new generation of collimators feature a design with embedded beam position monitors for a rapid alignment to the circulating beam [7], most of the LHC collimators do not have this feature. For the latter, the alignment relies on dedicated Beam Loss Monitoring (BLM) devices positioned outside the beam vacuum, immediately downstream from each collimator [4].

Collimator jaws are moved towards the beam with a step precision of 5 µm, and the BLMs are used to detect beam losses generated when halo particles impact the collimator jaws. The recorded losses are proportional to the amount of beam intercepted and are measured in units of Gy/s. Collimators are aligned with respect to a reference halo cut generated with the primary collimators. A collimator jaw is considered aligned when a movement produces a clear beam loss spike in the BLM [8]. The observation time to evaluate the quality of the signal and to assess if the spike corresponds to a correct alignment can vary from < 1 s to > 10 s depending on the machine conditions and beam properties. Aligning collimators with BLMs is referred to as the beam-based alignment (BBA), which involves aligning collimators one by one, by moving one jaw at a time towards the beam.

Before moving each jaw, the losses produced by the previous alignment must have decayed in order to decrease possible cross-talk effects between the collimators, whereby the BLM losses at a specific collimator are affected by the signal produced by other collimators around the LHC [9]. A complete alignment campaign at the LHC requires moving each collimator jaw several times, which can produce more than 1000 observation spikes. Therefore, improving the time needed to classify these spikes has a direct impact on the system's alignment time.

Semi-Automatic Beam-based Alignment

Since 2011, LHC collimators have been aligned using a semi-automatic procedure. This involves having the user to select the collimator to align, including the required settings and BLM threshold. The collimator will then automatically move towards the beam until the BLM losses exceed the threshold selected. At this point, the collimator automatically stops moving and the user must determine whether

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the collimator is aligned or not by classifying the loss spike recorded in the BLM signal.

Fully-Automatic Beam-based Alignment

The fully-automatic alignment was introduced and used in 2018 for all collimator alignments. As the name suggests, this fully-automates the entire procedure by automating the user's tasks in the semi-automatic alignment [10]. The three main user tasks have each been replaced with dedicated algorithms, such that:

- The collimator to align is automatically selected to avoid cross-talk (if any) [11].
- The BLM threshold (to stop the jaw movement during alignment) is automatically selected based on the real-time losses detected at the collimator [12].
- The BLM loss spikes are automatically classified using supervised machine learning into two classes; alignment spike or spurious spike [13]. This is possible by waiting a fixed duration of time to extract the features required for classification, from the BLM loss spike. Based on experience, the classification is set to wait 4 s at injection and 6 s at flat top.

These algorithms are developed as individual modules within the fully-automatic alignment software package [14], allowing for any improvements/upgrades. This paper focuses on the upgrade of the machine learning module.

ANALYSIS OF SEMI- AND FULLY-AUTOMATIC BBA

Data were collected from collimator alignments performed using the semi-automatic alignment software in 2016 injection commissioning and the latest parallel fully-automatic alignment software in 2018, during a dedicated beam test replicating injection commissioning conditions [15]. The logged data includes the alignment of 75 collimators in 2016 and 77 collimators in 2018, both at a frequency of 1 Hz. Table 1 lists the details of the two alignment campaigns, resulting in the fully-automatic procedure able to align the collimators at injection in one third of the time required by the semi-automatic one [16].

One can observe that the moving time of the collimator jaws during both campaigns is approximately 38 % of the total time (±10 % assuming a 0.5 s error on each jaw alignment due to the 1 Hz logging precision). This indicates that the fully-automatic alignment can be further sped up by decreasing the waiting time (62 %). The main contribution to the waiting time is the spike classification which is currently set to wait a fixed 4 s, at injection, for the BLM signal to decay before classifying it.

Decay Time Analysis

The actual time required for the BLM signal to decay at injection and flat top was analysed to identify possible gains in the overall alignment time by reducing the duration of the decay observation. The decay time was analysed for 1550 alignment spikes collected during 2016-2018 alignment campaigns, which includes 719 alignment spikes at injection and 831 at flat top.

Table 1: Details of two alignment campaigns at injection using the semi-automatic alignment in 2016 and the fully-automatic alignment in 2018.

	Semi-Automatic	**Fully-Automatic**
Collimators	75	77
Total time	2h 31m 59s	49m 17s
Moving time	58m 13s	18m 14s
Total alignments	1903	637
Moving time	38.3 %	38.0 %
Alignments/Coll	25.37	8.27

The decay rate is modelled as an exponentially falling distribution, with optimal losses achieved after 6 half-lives, as shown in Figure 1. As a result the optimal decay time is the latency required for the BLM losses to fall to 1/64 of the maximum value. Figure 2 displays the distribution of the decay latency at the two machine states such that the mean decay time is 0.61 s at injection and 2 s at flat top.

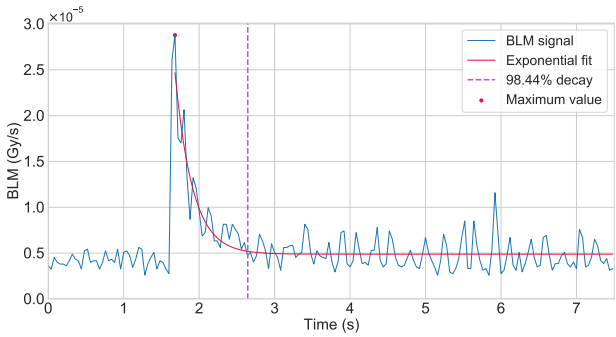


Figure 1: An example of a short decay observed in the BLM signal of an alignment spike at flat top.

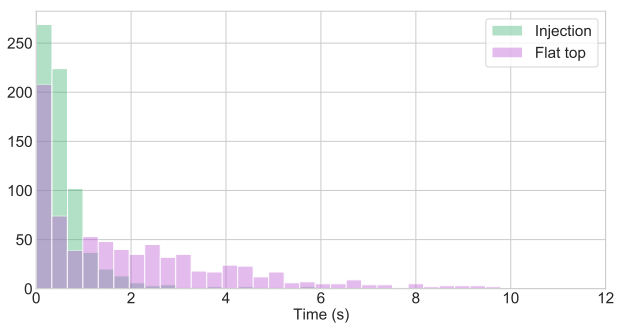


Figure 2: BLM signal decay time distributions at injection and flat top.

This highlights the fact that a dynamic adjustment of the observation time can speed up the overall alignment, as it is

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

not necessary to consistently wait 4 s at injection or 6 s at flat top. As an example, Figure 1 displays an alignment spike at flat top with a short decay that lasts 0.96 s. Moreover, this will allow for adjusting the observation window in the case of longer decays, although they do not happen frequently.

LSTM-RNN FOR SPIKE CLASSIFICATION

In order to continuously classify spikes in real-time and automatically detect their decay, the proposed solution is to train a Long Short-Term Memory (LSTM) - Recurrent Neural Network (RNN) [17]. This may enable the spike classification after a variable duration and minimize the waiting time of the automatic collimator alignment.

The entire data set gathered during 2016-2018 consists of 2973 samples collected when the BLM losses exceeded the predefined threshold, i.e. the moment when the observation of the BLM signal starts. The collected data samples were individually analysed and labelled into the classification classes; 1550 alignment spikes and 1423 spurious spikes. Each sample is a 7.5 s time series of length 188 and contains two signals; the BLM signal logged at a frequency of 25 Hz and the collimator jaw positions logged at a frequency of 1 Hz.

The input used to train the LSTM combines the two signals in each sample by scaling the BLM signal with the collimator position in sigma at the time the threshold was exceeded. Z-Score ($z = \frac{x-\mu}{\sigma}$) is then used as the normalization technique to re-scale the features such that they have the properties of a standard normal distribution.

The network architecture was developed using the deep learning library Keras [18], with TensorFlow [19] as the back-end, see Figure 3. It consists of two LSTM layers, followed by a dropout and dense layer. The model was trained over 75 epochs using an Adam optimizer [20] with a learning rate of $3e^{-4}$. Binary cross-entropy is used as the loss function.

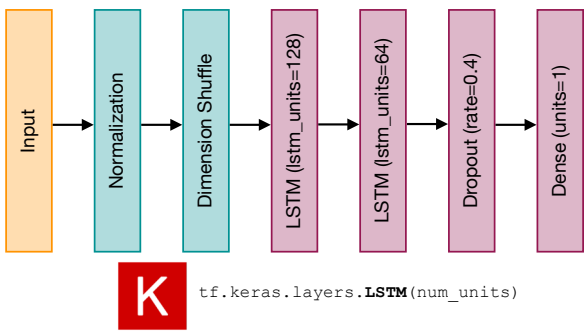


Figure 3: The LSTM network architecture. Two layers containing 128 and 64 hidden neurons, respectively, process the learning data produced by the preprocessing unit. Following is a dropout layer with rate of 0.4, then a dense layer with one neuron and sigmoid activation function.

The results of the two LSTM layers were outputted solely on the final time step, such that each layer generated a 2D array: Layer 1 output size: 188 × 128, Layer 2 output size: 64 × 1. Following this, the dropout layer outputs a 1D array of size 64, and finally the dense layer outputs the final probability used for spike classification.

In order to ensure the correct alignment of collimators, false detection of an alignment spike is more grievous than not detecting an alignment spike. Therefore, precision is used as the main performance metric to avoid false positives [13].

The results are collected over a 10-fold cross-validation randomly stratified 30 times, to handle lucky splits. The results are displayed in Figure 4, highlighting that the train and test loss curves stabilized with a minimal gap between the final values, thus indicating that a good fit has been found after 75 epochs. Overall, the model obtained an average precision of 94 % on the testing sets (and 94 % accuracy).

This precision was calculated by evaluating the classification probability at the end of the available sample, whereby a classification score with a probability larger than 50 % is classified as an alignment spike. Further analysis will aid to determine the best moment to predict the spike class and the ideal probability threshold, to possibly improve the model's precision.

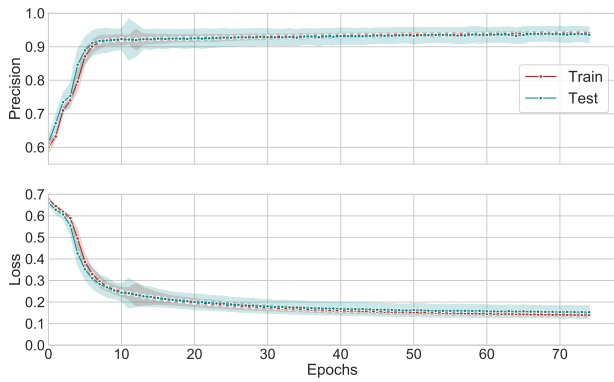


Figure 4: Loss and precision curves obtained by the LSTM on the training and testing data, in terms of mean and standard deviation.

Spike Classification Analysis

The trained LSTM model was used to continuously classify each sample at each time step, starting from the moment when the collimator stopped moving, until the end of the 7.5 s window.

An overview of the classification probabilities obtained for the two spike classes at injection is displayed in Figure 5. A clear distinction can be made between the spike classes at a latency of ~1.5 s, at which point the probability gradient for alignment spikes falls below 0.2. Taking a closer look at the classification probabilities of the two classes at a latency of 1.5 s in Figure 6, one can observe that there is no overlap at 80 % probability. Moreover, Figure 7 displays the latency required to obtain the maximum probability for each spike class. One can observe that the ~98% of spurious spikes obtain their maximum probability within the first 1.5 s, whilst

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

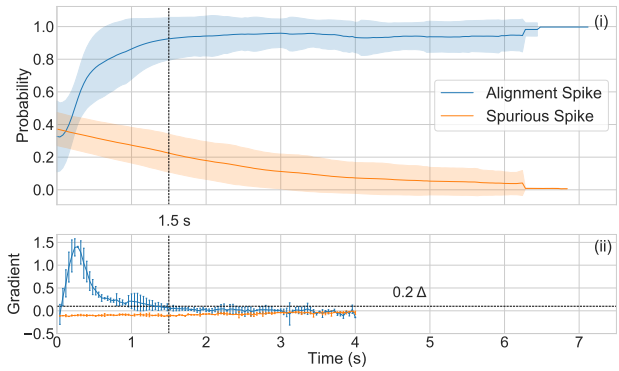


Figure 5: Spike classification results displayed as (i) the evolution of the classification probabilities in time for the two machine states, and (ii) the gradient of change in the classification probabilities. The two plots are displayed in terms of mean and standard deviation.

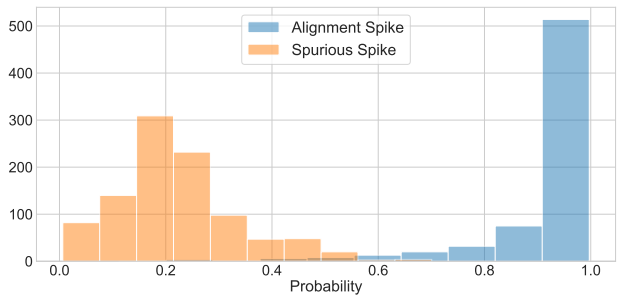


Figure 6: The probabilities obtained by the two spike classes at 1.5 s latency after the collimator stopped moving.

all spurious spikes remain below the 80 % classification probability threshold.

As a result, the LSTM can be set to make a classification once the probability gradient decreases below 0.2, which, as shown in Figure 5, occurs within a latency of ~1-1.5 s, at injection. If the probability at this point is below 80 %, then the BLM losses can be classified as a spurious spike, and the next alignment can begin. When the losses form an alignment spike, the next step is to determine the optimal BLM decay time to begin the next alignment. In this case an exponential function can be fit (as shown in Figure 1) to determine when ~98.5 % of the BLM signal decays (6 half-lives), which on average would have already decayed (mean of 0.61 s).

An analogous analysis was performed at flat top, showing similar results, i.e. classifications can rely on the 80 % probability threshold when the probability gradient decreases below 0.2, which at this machine state occurs within a latency of ~1.5-2 s.

Classification Results

Classifying the BLM signals as proposed by the presented analysis increases the classification precision on the data set to 98 % (with 90 % accuracy). In addition, the time taken for the LSTM to classify the data set into the two spike

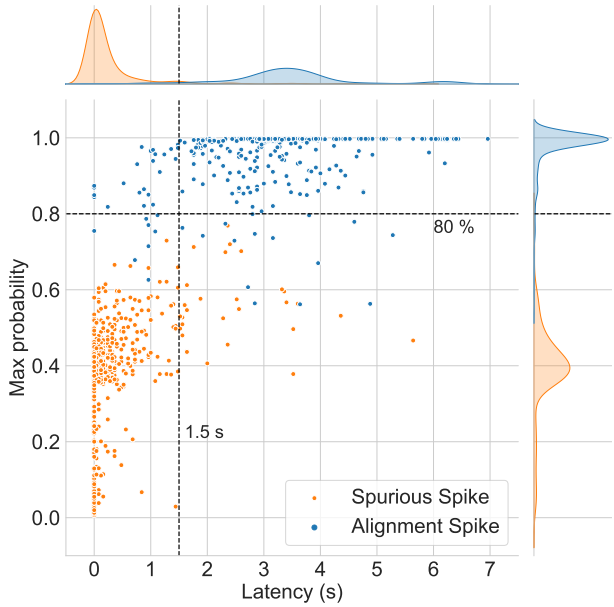


Figure 7: The distribution of the maximum probabilities achieved by the two spike classes and the required latency. The majority of alignment spikes obtain a probability above 80 % whereas all spurious spikes remain below.

classes resulted in a mean latency of 1.07 s at injection and 1.54 s at flat top, whereby 88 % of the observation spikes were classified within these times. Table 2 summarises the classification results at the two machine states, indicating a factor of 4 speed-up compared to the present implementation using supervised ML with fixed observation times.

Table 2: The latency results obtained when classifying the data set using the LSTM model. The classification was started at 1 s for injection and 1.5 s for flat top.

State	Start time	Mean	Stand. dev.	Max.
Injection	1 s	1.07 s	0.1 s	1.72 s
Flat top	1.5 s	1.54 s	0.06 s	2.04 s

Results with Ion Beams A data set was collated with 254 samples (192 alignment spikes, 62 spurious spikes) from the ion run in 2018 during the collimator alignment campaign in collisions. The mean decay time for the BLM losses to fall to 1/64 of the maximum value is 1.08 s.

An analogous analysis was performed with ion beams resulting in a similar environment for classifications, such that an 80 % probability threshold and 0.2 probability gradient after a minimum of 2 s latency, are ideal. The LSTM model trained on proton beams was used to classify this data set and obtained 97 % precision (and 87 % accuracy). The mean latency required to classify these samples is 2.08 s.

A summary of the results obtained at the different machine configurations analysed in this paper, is collected in Table 3.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

Table 3: Results obtained at different LHC configurations.

	Proton beams		Ion beams
	Injection	Flat top	Collisions
Decay time	~0.61 s	~2 s	~1.08 s
Prob. threshold	80 %	80 %	80 %
Prob. gradient	0.2	0.2	0.2
Class. latency	~1.07 s	~1.54 s	~2.08 s
Class. precision	**98 %**		**97 %**

THEORETICAL IMPROVEMENT OF ALIGNMENT TIME WITH LSTM

The time performance of the automatic BBA using supervised machine learning is displayed in Table 1, taking into consideration the following assumptions [11]:

- A clear alignment spike is achieved the first time the threshold is exceeded.
- Two clear alignment spikes are achieved after ~10 steps per jaw.
- The primary collimator was aligned in a previous alignment and both jaws achieved an alignment spike the first time the threshold is exceeded.

Table 4: Theoretical minimum time required to align a collimator [11].

Step	Action	Time (s)
1	Move both jaws to 4 mm	~8
2	Wait for losses to decay	x
3	Classification delay	1
4a	Align Left Jaw	2*(0.1 + x + 1)
4b	Align Right Jaw	2*(0.1 + x + 1)
5a	TCP before (Left Jaw)	0.1 + x + 1
5b	TCP before (Right Jaw)	0.1 + x + 1
6	TCP after (Left + Right)	2*(0.1 + x + 1)
Total	**17.8 + 9*x***	@Inj x >= 4 @FT x >= 6

The proposed LSTM model is capable of dynamically classifying alignment spikes of varying lengths in real-time. Therefore, this will decrease the time waiting for the losses to decay (x_1) to an average of 1.07 s at injection and 2 s for alignment spikes at flat top. This indicates that the theoretical minimum time required to align a single collimator at injection is 27.43 s, assuming that every spike is an alignment spike. Therefore, on average, aligning 79 collimators at injection would require a minimum of 36.1 mins.

On the other hand, if the collimator jaws encounter one spurious spike in Steps 1, 4a and 4b, then an additional 3 $*$ (0.1 + x_0 + 1) seconds are required per collimator, i.e. 1.07 s at injection and 1.54 s for spurious spikes at flat top. This results in an additional average of 6.51 s at injection, increasing the average time required in this case to 44.7 minutes.

Table 5: The average theoretical minimum time to sequentially align LHC collimators, calculated using Table 4.

Case studied	Supervised ML	LSTM-RNN
1 coll @Inj	53.8 s	27.43 s
+1 spurious spike	69.1 s	33.94 s
79 colls @Inj	70.84 mins	36.12 mins
+1 spurious spike	90.98 mins	44.69 mins
1 coll @FT	71.8 s	35.8 s
+1 spurious spike	93.1 s	43.72 s
79 colls @FT	94.53 mins	47.14 mins
+1 spurious spike	122.58 mins	57.56 mins

Table 5 summarises the theoretical minimum time required to sequentially align the collimator cases discussed, at injection and flat top. In 2018, the automatic alignment was upgraded to align the collimators in the two beams in parallel, resulting in 79 collimators aligned in 50 minutes at injection [15, 16]. Therefore the possible introduction of LSTM can theoretically align the collimators in ~24.56 minutes, speeding-up the automatic alignment by ~50 %.

CONCLUSION

The 123 LHC collimators are aligned automatically using supervised ML, provided by the latest fully-automatic software introduced in 2018. This paper analysed first-use performance of this software and identified a bottleneck caused by the fixed observation window used by the ML model to classify the BLM loss signal. This classification determines if the collimator jaws reached the correct alignment position.

In this paper a Long-Short Term Memory model was trained to continuously classify BLM signals. This allows classifying the losses into a spike class within 1-2 s of a collimator stopping its movement, once the rate of change in classification probabilities is below 0.2. Following each alignment spike classification, the suggestion is to fit an exponential function to the losses to determine whether the next alignment can begin. On average, the losses at injection would have already decayed (mean of 0.61 s), whereas decays at flat top may require a longer time (mean of 2 s).

This work allows for classifying BLM signals independent of whether the losses decayed or not, thus solving the bottleneck in the fully-automatic alignment process. Overall, this research could decrease the alignment time by ~50 %. The LSTM is readily available to be incorporated into the alignment software for testing during the LHC Run 3.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV040

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

REFERENCES

[1] O. Brüning, *et al.*, "LHC design report", vol. 1, CERN, Geneva, Switzerland, 2004.

[2] C. Bracco, "Commissioning Scenarios and Tests for the LHC Collimation system", Ph.D. thesis, Ecole Polytechnique Federale de Lausanne, 2009.

[3] G. Azzopardi, *et al.*, "LHC Collimation Controls System for Run III Operation", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Shanghai, China, 2021, paper THPV012, this conference.

[4] G. Valentino, "Fast automatic beam-based alignment of the LHC collimator jaws", Ph.D. thesis, Univeristy of Malta, 2014.

[5] M. Lamont, "The LHC from commissioning to operation", in *Proceedings of International Particle Accelerator Conference*, San Sebastian, Spain, paper MOYAA01, pp. 11–15, 2011.

[6] R. W. Aßmann, *et al.*, "Expected performance and beam-based optimization of the LHC collimation system", in *Proceedings of the European Particle Accelerator Conference*, Lucerne, Switzerland, pp. 1825 – 1827, 2004.

[7] G. Valentino, *et al.*, "First operational experience with embedded collimator BPMs in the LHC", in *Proceedings of the International Particle Accelerator Conference*, Busan, Korea, paper WEPMW034, 2016.

[8] E. B. Holzer, *et al.*, "Beam loss monitoring system for the LHC", in *IEEE Nuclear Science Symposium Conference Record, 2005*, vol. 2, pp. 1052–1056, 2005. doi:10.1109/NSSMIC.2005.1596433

[9] G. Azzopardi, B. M. Salvachua Ferrando, G. Valentino, "Data-driven cross-talk modeling of beam losses in LHC collimators", in *Physical Review Accelerators and Beams*, vol. 22, no. 8, 2019.

[10] G. Azzopardi, G. Valentino, B. M. Salvachua Ferrando, S. Redaelli, A. Muscat, "Software Architecture for Automatic LHC Collimator Alignment using Machine Learning", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, New York, NY, USA, paper MOCPL04, pp. 78–85, 2019.

[11] G. Azzopardi, "Automation of the LHC Collimator Beam-Based Alignment Procedure for Nominal Operation", Ph.D. thesis, Univeristy of Malta, 2019.

[12] G. Azzopardi, *et al.*, "Automatic Beam Loss Threshold Selection for LHC Collimator Alignment", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, New York, NY, USA, paper MOPHA010, pp. 20–213, 2019.

[13] G. Azzopardi, G. Valentino, A. Muscat, B. M. Salvachua Ferrando, "Automatic spike detection in beam loss signals for LHC collimator alignment", in *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 934, pp. 10–18, 2019.

[14] G. Azzopardi, "The Automatic LHC Collimator Beam-Based Alignment Software Package", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Shanghai, China, paper WEPV016, this conference, 2021.

[15] G. Azzopardi, B. M. Salvachua Ferrando, G. Valentino, S. Redaelli, "Operational Results on the Fully-Automatic LHC Collimator Alignment", in *Physical Review Accelerators and Beams*, vol. 22, pp. 1208–1211, 2019.

[16] G. Azzopardi, B. M. Salvachua Ferrando, G. Valentino, S. Readelli, A. Muscat, "Operational Results of LHC collimator alignment using machine learning", in *Proceedings of the International Particle Accelerator Conference 2019*, Melbourne, Australia, 2019.

[17] S. Hochreiter, J. Schmidhuber, "Long short-term memory", in *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] F. Chollet, *et al.*, "Keras", 2015. Available at: https://github.com/fchollet/keras.

[19] M. Abadi, *et al.*, "Tensorflow: A system for large-scale machine learning", in *12th Symposium on Operating Systems Design and Implementation*, 2016.

[20] D. P. Kingma, J. L. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980, 2014.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

INNOVATIVE METHODOLOGY DEDICATED TO THE CERN LHC CRYOGENIC VALVES BASED ON MODERN ALGORITHM FOR FAULT DETECTION AND PREDICTIVE DIAGNOSTICS.

A. Amodio, P. Arpaia, Y. Donon, F. Gargiulo, L. Iodice, M. Pezzetti, CERN, Geneva, Switzerland

Abstract

The European Organization for Nuclear Research (CERN) cryogenic infrastructure is composed of many equipment, among them there are the cryogenic valves widely used in the Large Hadron Collider (LHC) cryogenic facility. At present time, diagnostic solutions that can be integrated into the process control systems, capable to identify leak failures in valves bellows, are not available. The authors goal has been the development of a system that allows the detection of helium leaking valves during normal operation using available data extracted from the control system. The design constraints (inaccessibility to the plants, variety of valve models used) has driven the development towards a solution integrated in the monitoring systems in use, not requiring manual interventions. The methodology presented in this article is based on the extraction of distinctive features (analyzing the data in time and frequency domain) which are exploited in the next phase of machine learning. The aim is to identify a list of candidate valves with a high probability of helium leakage. The proposed methodology, which is at very early stage now, with the evolution of the data set and the iterative approach for the test phase presented in the last paragraph, is aiming toward a cryogenic valves targeted maintenance in the LHC cryogenic accelerator system.

INTRODUCTION

The maintenance purpose is to reduce, as far as possible, the occurrence of undesirable events and, consequently, the corrective maintenance interventions. The maintenance campaign of large accelerator systems, such as the LHC at CERN, represents an important factor in terms of financial and manpower resources. At CERN, a large fraction of the cryogenic installation and its control systems are located in areas inaccessible during physic run campaigns. Due to the high complexity of the accelerator, the cryogenic system needs high levels of reliability for its operations. [1]. The cryogenic valves are widely used in the LHC cryogenic facility. The design constraints, such as the inaccessibility to the plants and the variety of valve models used, have driven the development of an integrated solution in the monitoring systems in use, not requiring manual interventions. The authors motivation has been the development of a system that allows the detection of helium leakage during valves operations using available data. In the past, several diagnostic approaches have been developed concerning compressors, electrical motors and cryogenic instrumentation [2, 3]. In literature, diagnostic solutions that can be integrated into the process control systems using only the available data, to identify leak failures in valves bellows, are not available at this time. Although the solution proposed in [4] based on Support Vector Machine (SVM) reaches a very interesting level of accuracy (97 %), it unfortunately requires the use of vibration sensors making this method inapplicable in contexts where the valves are numerous and difficult to access. The state-of-the-art solutions for failure prediction in control valves, cannot fit the context whose constraints are described.

PROBLEM OF CRYOGENIC VALVE BELLOWS LEAKAGE

The helium used in cryogenic systems can, due to its physical characteristic, escape through micro-cracks originated from valve bellows movements after years of operations. The main purpose of the presented work is the development of an innovative tool for maintenance diagnostic. The described algorithm produces a list of designated valves that could potentially present the helium leakage problem. The selected valves are investigated, by cryogenic operators, using local helium sniffing devices and if the leak is validated, a mechanical repair action is undertaken. The cryogenic system uses several valve models for the regulation and the control of cryogenic liquefied gases, being able to be fully functional both at temperatures as low as 1.9 K and at various pressures required by the cryogenic process. The system presented below is focused on these control valves of the cryogenic types (see Fig. 1) fabricated in stainless steel (AISI 316L). The control valve model used at CERN is driven by a Siemens Sipart PS2® positioner which gets a setpoint by the industrial Profibus® PA or by a 4-20 mA current signal (supporting HART® protocol). The control valve has a chamber divided into two parts by a diaphragm.

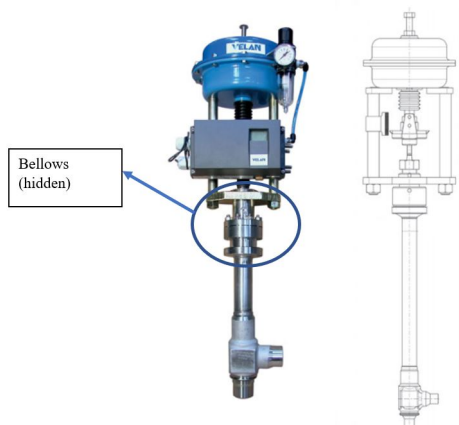


Figure 1: Example of LHC Cryogenic control valve [5].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The desired state of the valve is achieved by varying the air pressure in the two areas of the chamber. The pressure difference moves the diaphragm, and the stem tuning the valve opening. The valves are sealed to air using metallic bellows forming a cylinder shape junction between the fixed and movable part of the valve. The bellows shape is designed in order to be flexible and to follow the stem actions. The authors have analyzed a subset of 174 out of the 1357 control valves.

PRIOR KNOWLEDGE MODEL

The valve failures, causing helium leakage, are mainly related to the metallic bellows fatigue due to the workloads within the tens of thousands of elongations cycles [6].The study turns into a binary classification problem in which the fatigue phenomena is used to identify if the valve is close to a failure caused by stress. The fatigue phenomena is described using the Wöhler diagram (see Fig. 2). It represents the number of cycles at which a breakage is expected to happen. According the diagram, the relationship between stress σ and critical number of the cycles N is:

$$\sigma^M \cdot N = constant \tag{1}$$

where M is a constant.

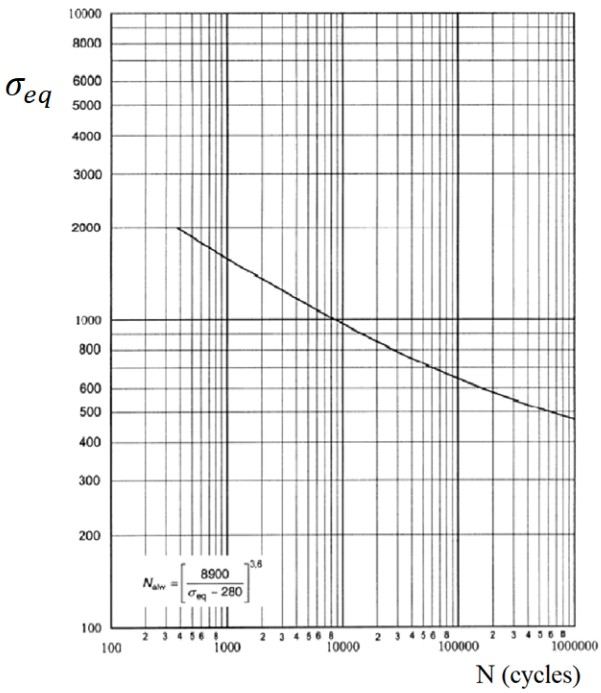


Figure 2: $\sigma - N$ diagram of stainless-steel bellows [7].

During normal operation, the mechanical effort on the component is characterized by different values of stress intensity. To combine them and estimate the general cumulative damage of the material, Palmgren-Miner's rule is used [8].

$$\sum_i \frac{n_i}{N_i} = damage\ condition\ parameter \tag{2}$$

where n_i and N_i are the number of performed and the critical cycles of the $i-th$ value of stress applied to the bellows. The damage condition parameter indicates an estimation of the cumulative damage and is used to compare the health status of the valves. Values under the unit mean that the fatigue limit is not reached.

DATA COLLECTION

The history of the valves position is extracted from the logging files, and the parameters reported in the Table 1 were gathered for each valve. The dataset is built from log files using a Python® script and the CERN/TIMBER® application querying of the CERN Accelerator Logging Service. The extracted information were then organized as a vector for each valve. The history covers the interval of time between 2008 and 2019.

Table 1: SIPART PS2 Positioner® Parameters

STRKS	Number of complete strokes from 0% up to 100% and back
CHDIR	Number of times a change of direction has occurred
HOURS	Number of hours worked since initialization
SSUP	It defines the upward variation within which the valve in the Slow Step zone is operated. In this condition, the actuators are piloted by a PWM signal to avoid overshoots.
SSDN	Similarly to the SSUP parameter but with regard to the downward movements.

After having studied the problem from a physical point of view and collected the data, the authors have divided the procedure into two main steps:i) Pre-processing phase where distinctive features are extracted by considering breaking phenomena; ii) Machine Learning (ML) model training, for the classification of broken valves. While the leak problem is tackled, the reliability of the ML model obtained will improve because of the growing number of broken valves dataset.

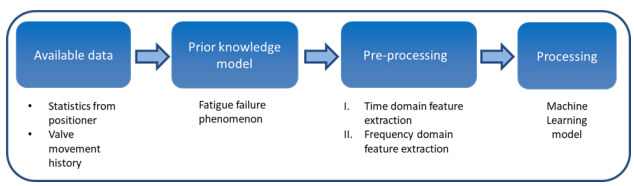


Figure 3: Representation of algorithm approach development.

PRE-PROCESSING FOR FEATURES EXTRACTION

Starting from the vector of the valve position over time, two features were extracted to be used in the subsequent ML model training. The extraction of these two features takes place in the two phases explained below:

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

Time Domain

This phase aims to identify which valve bellows had the most stressful regulation due to the widest elongation and compression. Considering the bellows as a spring, the Hook's law can be used to find a relation between the amplitude and the force applied to compress or stretch the bellows (and indeed stress):

$$\sigma = \frac{-k \cdot \Delta L}{S} \Rightarrow N \propto 1/\Delta L^M \tag{3}$$

where k is the elastic constant of the material, ΔL is the excursion performed by the bellows and S is its section area. In according to fatigue theory, larger movements of the valves (opening or closing) drastically reduce the lifetime of the bellows. The evaluation of the correct amplitude is mandatory because of the exponential relationship between the critical number of cycles and the amplitude of the movement. The series of movements are filtered and then processed with logical operations to obtain the extremes mono directional variation. This operations not only allow to reduce the vector size by discarding all the value between two extremes, but they also reduce the computational effort for all the subsequent operations. The little changes in directions are removed using the local Gaussian smoothing. At the end, only the extremes of the wide variations are extracted. The purpose is to build a matrix from the dataset in order to have a synthetic representation of the movement history. The Stochastic matrix is a square matrix in which the rows and the columns represent respectively the initial and the final position of the movement. The matrix is filled with the number of movements relative to the row-column pair. A mask matrix is then adopted in order to assign a weight to the excursions according to the change of position importance (more or less burdensome). According to Palmgren-Miner's rule, the mask coefficients are the reverse of the maximum number of expected cycles and are obtained by the Wohler diagram of stainless steel bellows. At CERN, the information from broken valve dataset is extracted in order to improve the analysis with experimental data. The dataset of broken bellows valve and healthy valve are considered, and a stochastic matrix is extracted and calculated for each of them. All the stochastic matrixes are weighted with the coefficient of the mask and are clustered using an unsupervised classifier (ML technique in which the users do not need to intervene manually to label the dataset model) to identify for each coefficient of the matrix a centroid of the broken valve set. All these centroids are collected in a matrix and each value is divided by the mean value of stochastic matrix of the broken valve dataset. The unsupervised classifier used is $K-means$. The outcoming matrix is used as weights matrix for the analysis of each new checked valve. A final output parameter is extracted for each valve to test. The parameter is the sum of all the coefficients of the matrix obtained by multiplying the coefficient of the stochastic matrix and the second weight matrix. This parameter, according to previous formulas, is proportional to the estimation of the accumulated bellows damage.

Frequency Domain

The second phase is based on spectral analysis performed to evaluate the dynamic of the movements. The time series data extracted from log file are non-uniform sampled. Before passing to the frequency domain, an interpolation must be performed. The interpolation must fill the missing values discarded during the the storing in log files (due to avoiding the replication of similar values in log files). Considering a common time base for all valves, a previous neighbor interpolation to replicate the non-sampled values is performed. Other interpolation approaches are avoided in order to do not alter the original values. The spectral analysis is performed by calculating the energy of the envelope of the module of the Fast Fourier Transform (FFT) of valve trace movements. The extracted feature is the ratio between the energy of the last quarter of the frequency spectrum and the total energy.

MACHINE LEARNING PERFORMANCE MEASURES

To assess the quality of the classification model, different performance indices were considered. The accuracy index (explained below), when the size of the minor class represents only a small percentage of the data set size, is not suitable because the minority class has very little influence on accuracy. The results of binary classification can be defined in four different values: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Table 2: Confusion Matrix for Binary Classification

		Predicted class	
		Positive	Negative
True Class	Positive	TP	FN
	Negative	FP	TN

By these values the confusion matrix was built as shown in Table 2. Different quality indexes can be calculated:

$$Accuracy\ (ACC) := \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N}$$

$$Recall\ (REC) := TP_{rate} = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$Precision\ (PREC) := \frac{TP}{TP + FP}$$

$$False\ Positive\ Rate\ (FRP) := \frac{FP}{TN + FP}$$

where P refers to the total number of broken valves and N refers to the total number of healthy valves.
The TPR and the PRECISION values are summarized in the F-score parameter defined as:

$$F = 2\frac{Precision \cdot TPR}{Precision + TPR}$$

These indices are useful to give a measure of performance and to make comparison between the models.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

EXPERIMENTAL RESULTS

In the processing phase, different binary classification models were explored [9]. The choice of the model is driven by the performance indices explained in the section "Machine Learning Performance measures". The input dataset is composed of the observations collected for each valve, whose features are the elements in Table 1 and the time and frequency domain features extracted as described above. In the first experiment 174 valves, of which 8 were broken, were used as input data (training set), . The choice of the valves is driven by the similar mechanical characteristics, in order to make them comparable using Wöhler diagram. Due to the small dataset, the K-fold cross-validation is chosen with k=8 to have at least 1 broken valve for each validation subset. To train the model, only the predictors with low correlation are chosen. The selected predictors were the two features extracted in pre-processing step, CHDIR and STRKS. The model was trained with these features obtaining as best result, in terms of accuracy, the "Gaussian SVM" with an accuracy of 96.6 %. The broken valves recognized were two out of eight, with a TPR of 25.0 %. Even though Kernel Naïve Bayes model has an accuracy lower and equal to 94.8 %, it had a TPR for the broken class of 75.0 % identifying six out of eight broken valves. The F-score is equal to 57.1 %. Good results were obtained although the dataset is highly unbalanced with a TNR of 95.8 % and a TPR of 75.0 %. Different techniques were analyzed to overcome the disproportion between the positive and negative class [10].

Cost-sensitive Learning Technique

Two different costs for the wrong classification are assigned. The aim is to penalize the wrong classification of positives samples more than the wrong classification of negatives samples, hence a higher cost to FN is applied [11]. The CFN denotes the cost of predicting negative instead of

Table 3: Misclassification Costs Matrix

		Predicted class	
		Positive	Negative
True Class	Positive	0	CFN
	Negative	CFP	0

positive, and CFP the cost of predicting positive instead of negative. To move the attention of the model to the broken valves, a script was written to maximize the F-score varying the CostRatio and, at the same time, minimize the cost function [12]. Different models achieved good results as shown in Table 4. In this case, the best solution is the Quadratic Discriminant model, which accomplished a TPR=87.5 % recognizing 7 out of 8 broken valves and a TNR=95.8 %.

Resampling Techniques

Operating before the training, a dataset already balanced in the classes has been provided to the Classification Learner App®. There are two different techniques :

Table 4: TPR and TNR with a different CostRatio [9].

	TPR(%)	TNR (%)	CR	New TPR (%)	New TNR (%)
Linear Discriminant	50	98.2	6	62.5	96.4
Quadratic Discriminant	75.0	96.4	3	87.5	95.8
Subspace Discriminant	50	98.2	23	75	94.0
Linear SVM	25.0	98.2	9	75.0	96.4
Medium SVM	25.0	100	23	62.5	95.2
Coarse Gaussian SVM	0	99.4	18	75	93.4
Kernel Naive Bayes	75.0	95.8	9	87.5	92.2

- "Undersampling" which purpose is the reduction of the samples in the majority class to balance the data. The technique was not performed due to the small number of samples in the minority class.

- "Oversampling" which purpose is the increment of the samples of the minority class to balance the data. This result can be achieved either randomly or with synthetic data. These techniques can be then adopted, but overfitting risk need to be handled.

Synthetic data are added to the dataset by means of the following oversampling techniques:

- Synthetic Minority Over-sampling Technique (SMOTE): New synthetic samples are obtained by selecting randomly one of the $K - nearestneighbors$ of a sample casually picked from the minority class. Then, through a random number $\delta \in [0, 1]$, the point is placed along the line between the selected minority class sample and the nearest neighbor chosen [10]:

$$X_j^{SMOTE} = x_i(\tilde{x}_{i,k}^{Knn} - x_i) * \delta_j \quad (4)$$

where $\delta \in [0, 1]$ is a random number.

In this case study, an oversampling of 8 broken valves

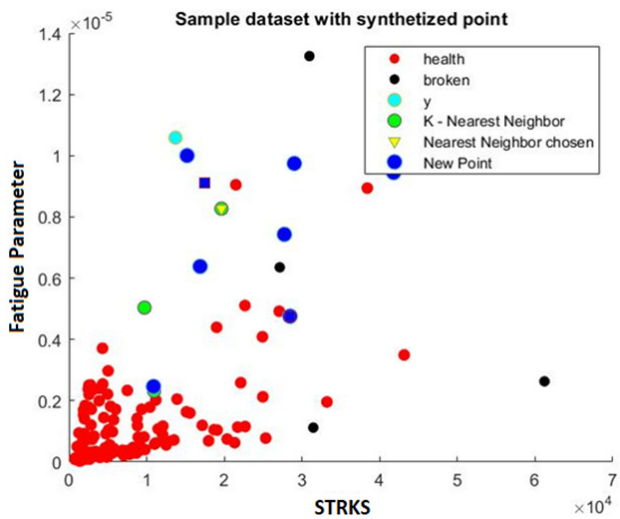


Figure 4: Synthetic data SMOTE.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

was processed, in order to prevent the model from being performed only on the synthetic data, leading to a completely distorted model. Factor K=3 and a 16-cross validation are chosen obtaining as best result the "Quadratic Discriminant" with an accuracy of 95.1 % with the confusion matrix in Table 5.

Table 5: SMOTE Confusion Matrix

		Predicted class	
		Positive	Negative
True Class	Positive	14	2
	Negative	7	159

Cross-validation and Overfitting

If the oversampling of the minority class is done before cross-validation, the oversampling can lead to overfitting. Because in the Smote algorithm δ_i could be equal to 0 or equal to 1, if the oversampling is applied before the cross-validation, one or more samples could be equal to the original dataset points [13]. In Fig. 5 the same minority samples (in blue) are in both validation and training set. The overfitting can be avoided if in each iteration: i) dataset is divided into validation and training set, then ii) SMOTE algorithm is applied only to the training dataset and iii) performance measures are evaluated with the validation set [13] as shown in the Fig. 6. Applying this structure, there has been a degradation in the performances of the SMOTE algorithm with a TPR=75.0 % recognizing 6 out of 8 broken valves.

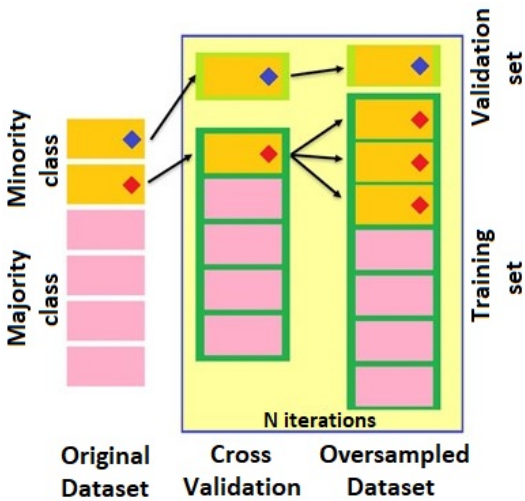


Figure 5: Oversampling and Cross-validation.

ITERATIVE APPROACH FOR THE TEST PHASE

Since this is a novel study of the helium leakage of the cryogenic valves, the test phase is conducted by applying the ML models introduced by authors, to the valves of a certain sector of the LHC. The valves are then manually checked for damages. This approach allows to evaluate the performance of the model and improve it at each step. Due to the low number of broken valves, one of the first algorithm attempts, then rejected, presented the problem of overfitting as the performance in the test phase was much worse than in the training phase. For this reason, the authors decided to consider several dataset manipulation techniques and ML solutions towards the highest level of performance. The authors are confident that the combination of the increasing number of broken valves and the iterative approach to the test phase will lead to a more reliable model for the correct recognition of broken valves.

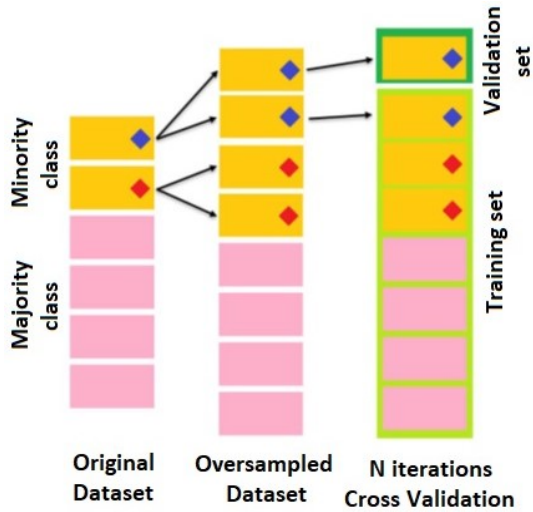


Figure 6: Cross-validation and Oversampling.

CONCLUSION

The authors present in this paper an innovative solution for fault detection of cryogenic valves bellows installed in the CERN LHC accelerator. The development was focused to produce a solution able to be easily applicable to the data from the existing cryogenic control system. The solution consists of a pre-processing step in which a features extraction is performed and a ML phase for the data drive modelling. The dataset, for the training and validation step, is composed of recorded data characterized of 174 valves by different features. Several classifiers were validated and best performances, exploiting the oversampling of broken valves, were reached by means of Quadratic Discriminant. The Quadratic Discriminant model has accomplished good performances both with oversampling and cost-sensitive learning technique. The presented results were obtained using an unbalanced dataset of heterogeneous types of bellows-sealed control valves. The work presented in this manuscript will constantly evolve when more dataset become available to improve the training and validation steps and perform the iterative test step.

REFERENCES

[1] L. Serio, "Machine learning-based system for the availabil-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV041

ity and reliability assessment and management of critical infrastructures (caso)," 2019.

[2] L. Serio *et al.*, "CERN experience and strategy for the maintenance of cryogenic plants and distribution systems," *IOP Conference Series: Materials Science and Engineering*, vol. 101, p. 012 140, Dec. 2015. DOI: 10.1088/1757-899x/101/1/012140. https://doi.org/10.1088/1757-899x/101/1/012140

[3] P. Arpaia *et al.*, "Fault detection on fluid machinery using hidden markov models," *Measurement*, vol. 151, p. 107 126, 2020.

[4] S. K. Venkata and S. Rao, "Fault detection of a flow control valve using vibration analysis and support vector machine," *Electronics*, vol. 8, no. 10, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8101062. https://www.mdpi.com/2079-9292/8/10/1062

[5] Velan. "Velan website." (accessed: 02.11.2020), https://www.velan.com/

[6] J. Fydrych and G. Consogno, "A maintenance strategy for a multi-valve cryogenic distribution system," *IOP Conference Series: Materials Science and Engineering*, vol. 278, p. 012 014, Dec. 2017. DOI: 10.1088/1757-899x/278/1/012014. https://doi.org/10.1088/1757-899x/278/1/012014

[7] AFNOR. "NF EN 14917 A1." (accessed: 02.11.2020), https://cds.cern.ch/record/2234138/files/NF-EN-14917+A1.pdf? French standard: Metal bellows expansion joints for pressure applications.

[8] M. A. Miner, "Cumulative damage in fatigue," 1945.

[9] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009. DOI: 10.1109/TKDE.2008.239.

[11] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.04.009. https://www.sciencedirect.com/science/article/pii/S0031320307001835

[12] P. Arpaia, M. Girone, D. Maisto, C. Manna, and M. Pezzetti, "Generalized extremal optimization of predictive maintenance to enhance monitoring of large experimental systems,"

[13] S. Mishra, "Handling imbalanced data: Smote vs. random undersampling," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 8, 2017.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

EVOLUTION OF THE CERN BEAM INSTRUMENTATION OFFLINE ANALYSIS FRAMEWORK (OAF)

A. Samantas, M. Gonzalez-Berges, J-J Gras, S. Zanzottera, CERN, Geneva, Switzerland

Abstract

The CERN accelerators require a large number of instruments, measuring different beam parameters like position, losses, current etc. The instruments' associated electronics and software also produce information about their status. All these data are stored in a database for later analysis. The Beam Instrumentation group developed the Offline Analysis Framework some years ago to regularly and systematically analyze these data. The framework has been successfully used for nearly 100 different analyses that ran regularly by the end of the LHC run 2. Currently it is being updated for run 3 with modern and efficient tools to improve its usability and data analysis power. In particular, the architecture has been reviewed to have a modular design to facilitate the maintenance and the future evolution of the tool. A new web based application is being developed to facilitate the users' access both to online configuration and to results. This paper will describe all these evolutions and outline possible lines of work for further improvements.

INTRODUCTION

Several thousand instruments are installed in the CERN accelerator complex to measure beam parameters (e.g. position, losses, intensity, etc). Table 1 gives an approximate overview of the types of instrument per accelerator. Each instrument is typically composed of a monitor that can be inserted in the beam pipe or installed outside, an analogue/digital electronics system and a software layer.

The complex itself spans over several kilometres, with most of the accelerators installed underground. The regular operation of these instruments is a major challenge. They need to be available for the run periods and their performance has to be guaranteed.

Table 1: Approximate Number of Main Types of Instruments per Type and Accelerator

	LHC	SPS	PS complex	Other
Position	1300	300	300	50
Losses	4200	10	50	30
Intensity	20	10	80	10
Profile	40	10	80	30

The Offline Analysis Framework (OAF) [1,2] was developed some years ago to deal with this challenge. The instruments produce beam physics data as well as status information. Both sets of data are used to monitor the instrument performance and its evolution through time. Our final aim is to fine tune the instruments' performance and introduce predictive maintenance on their mechanics and electronics parts.

CURRENT USAGE

All these instruments measure, regularly or on demand, the different beam observables. Then, they send these values, together with relevant status and setting registers, to the CERN Control Room for the real time operation of the machines and into a centralized logging database (NXCALS) [3] for future offline analysis (Fig. 1).

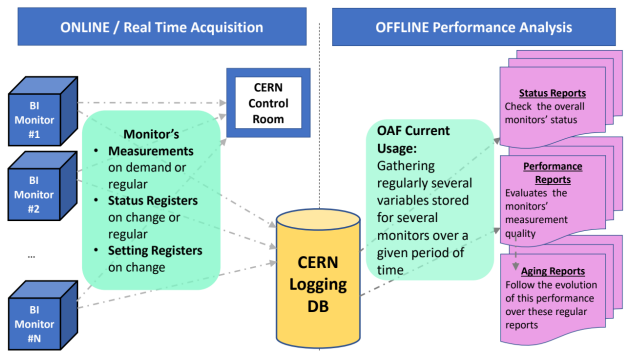


Figure 1: Standard usage workflow.

All this logged information contains too much data to be analysed properly manually. So, in 2013, we decided to develop the Offline Analysis Framework [1] in order to regularly monitor the records stored and automatically produce 3 kinds of reports:

- Status reports that focus on the state of the instrument and monitor humidity, temperatures, logging frequency and more, raising alarms whenever necessary.
- Performance reports will monitor and assess the quality (accuracy, resolution, stability) via device comparisons or regular calibration sequences.
- Aging or long term evolution reports will survey the evolution of the quality evaluations made in the daily performance reports.

Most of these analyses can be derived from the raw DB records via standard data treatment and plotting functionalities directly supported by OAF. However, some analyses require ad-hoc computations or specific plots. To cover these non-standard requirements, OAF offers the possibility, whenever necessary (i.e. when the need is not covered by the build-in OAF features), to add expert python code to this specific analysis that will be

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

executed at the end of the standard OAF process with a direct and simple access to:

- The extracted raw DB data and all subsequent data produced by the OAF associated treatments.
- The OAF programmable interface which allows the expert graphs, tables and alarms produced to be included in the reports.

ARCHITECTURE REDESIGN

Although stable and capable of running reliably for years, the original software architecture of OAF had some flaws that made working with its codebase difficult.

For example, adding new types of analyses to be run on the collected data was very complex: it required the original author to write lots of code, and it was not modular, meaning that no code could be reused from one analysis to the other. As a result, over the years the source inflated due to the code duplication, and became very hard to maintain and develop further.

An additional issue we found was that users did not find the user interface understandable enough to use it, and required constant assistance from the original author to use the system.

Recently, CERN adopted NXCALS as its central logging database, forcing OAF to adapt to this new interface. Due to the need to perform this upgrade, we took the opportunity to review the entire system, and we decided to perform a nearly complete redesign. The aim of this redesign was:

- Make OAF modular and maintainable.
- Make OAF easier to use for the end users.
- Make the creation of new analysis types easier.
- Review which of the existing analysis were still being used, and which ones could be dropped.
- Clean up and simplify the codebase heavily.
- Strongly improve the performance on some of the analysis.

As a plus, we also agreed that the output of OAF could be improved by adding more output mediums. From the original PDF-over-email report (Fig. 2), we decided to add several others: a web application, a raw data file, and we plan to add Python notebooks as well, to allow users to explore the results further (Fig. 3).

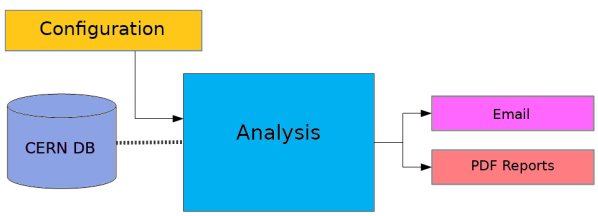


Figure 2: Original architecture of OAF.

One more incentive to the redesign was that the original OAF has been developed in Python 2. This fact meant that a full rewrite was not necessary, but large chunks of the old code could be kept in the new architecture with minimal or no changes.

During the redesign, we identified several logic units and split the codebase into meaningful modules, which were then reconnected over a more explicit API. This process proved to be very helpful to the codebase simplification goal we identified earlier.

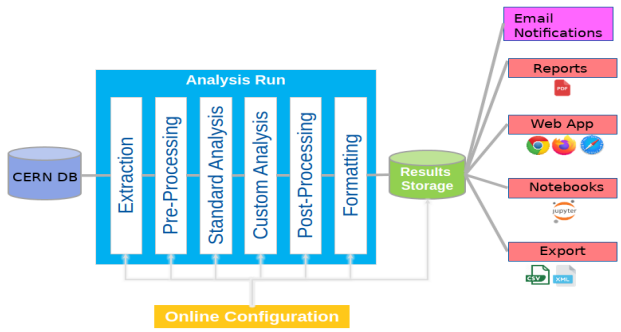


Figure 3: OAF architecture after the redesign.

In between the last step of the analysis and the output, we also introduced the concept of *results storage*, a temporary store for the processed data. This storage was critical to make our web interface capable of showing the analysis' results without having to re-run the analysis itself from the original logged data. This storage, though, also represented a challenge, as we wanted to avoid data duplication with the database as much as possible. We finally decided to use HDF5 files for storage, due to the format's properties and its capability to efficiently store complex data structures.

Along with the intermediate storage, we also developed a small utility library to act as an interface between the raw files and the Python code, called oaf-commons, which could be shared between the proper OAF pipeline and the web application without incurring in more code duplication.

As a result of the redesign the runtime architecture completely changed. This can be seen in the Figure 4. The users interact with the system through a web application that will be described in the next paragraph. For complex analysis where custom code is needed it will be placed directly in Gitlab. The OAF engine gets from Gitlab all the configuration and the custom code. It queries the logging database and produces results in a distributed file system. Then, the web application can access these results and display them.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

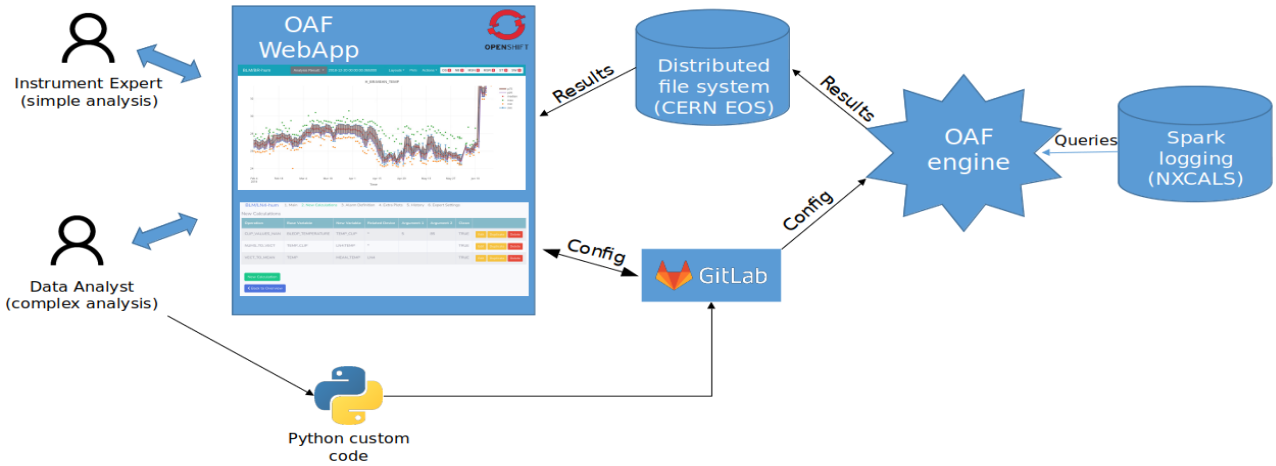


Figure 4: Runtime architecture.

OAF WEB APPLICATION

The OAF Web Application (Webapp), is a web based application in Python using Django Framework for both visualization of BI data results and analysis configuration.

Visualization of Results

All the HDF5 files that contain the different instrument analysis results are saved in a specific directory. In the Webapp, a list of different instrument analysis where we can browse between results from different dates is generated. By default one plot is visualized but there is also the option to visualize two or four plots the same time (Fig. 5).

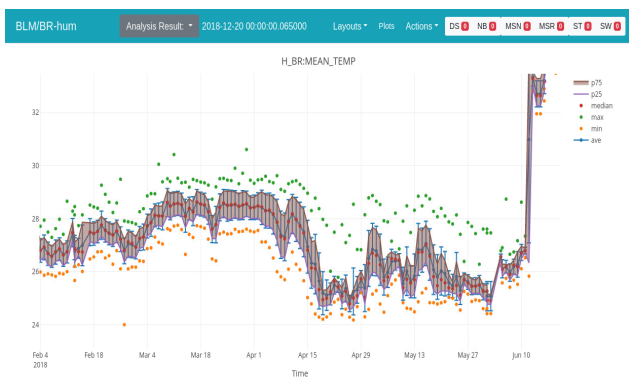


Figure 5: OAF Webapp instrument analysis example.

There are several alarms, various plot types and categories. OAF can raise an alarm in the following instances:

- MISSING: Alarm raised in case of missing entities.
- NB_REC: Alarm raised when number of records is not as expected.
- DISCRETE: Alarm raised if values do not belong to the given list of values.
- MEASURE: Alarm raised if values do not remain within given range.
- STATUS_BIT: Alarm raised if the state of status bit register does not correspond to the nominal state of each bit.
- SWITCH: Alarm is raised each time a value is changed.

The following plot categories that are already implemented are:

- Default Plots: Input data used for the analysis coming from NXCALS.
- Extra Plots: Produce more complex plots and tables combining default plots or custom ones on demand.
- Historical Plots: Show evolution of parameters across different analysis executions.
- Admin Plots: Monitoring and analysis of OAF itself.
- Info Plots: Summary of alarms and other info.

Each of these categories can produce different types of plots. These types can be value over time, value over value, Histogram, Num-box, Tables and many more.

Quite often we need to transform the displayed results in order to understand better the measurements. The most common conversion types (statistic calculations, FFTs, KDE) have been identified and are made available to the user. Different plot features are already implemented, providing like this a deeper, more specific and accurate analysis according to the users' needs.

An overview of all the alarms that were detected the last 24h in each of the available instrument analyses is provided in a table (Fig. 6).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Instrument	Analysis	Total Alarms	Newest Result	Comment
filter data...				
BLOG	AD	0	2018-06-15 00:01:17.333000	
BLOG	PSB		2018-06-15 00:00:00.065000	
BLOG	CPS-ana	2	2018-06-15 23:40:01.900000	
BLOG	SPS		2018-06-15 00:00:03.735000	
BLOG	LEI	0	2018-06-15 00:00:00.455000	
BOFSU	LHC-survey	0	2021-04-30 00:00:00.000363275	
BCT	LHC-B1-cmp	0	2021-02-19 00:01:00	
BCT	SPS-safety	0	2018-06-15 00:00:03.735000	
BCT	LHC-fast-ldm	0	2018-05-06 23:40:00	
BCT	L2-trans	0	2018-11-11 23:00:00.065000	
BCT	L3-trans	0	2018-11-10 00:00:00.455000	
BCT	LHC-B2-cmp		2021-03-23 00:01:00	
BCT	LHC-calib	0	2018-06-24 00:00:00	

Figure 6: Overview alarm table.

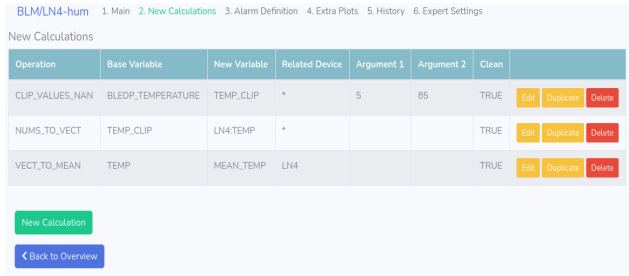


Figure 7: OAF Webapp configuration wizard.

Analysis Configuration

Apart from the visualization of results in the Webapp one can define the configuration of the analysis. Until today the user had to configure excel files without hints and error control. In the Webapp version we provide a user friendly interface guiding the user through a wizard.

Configuration is achieved by the modification of different tables (Fig. 7) that are stored in an internal database. This allows the configuration of many features of the framework. The instrumentation experts do not have to provide any code for a common usage case. The different table categories are the following:

- Main: General information.
- New Calculations: Declaration of new variables based on extracted ones.
- Alarm Definition: The criteria for the alarms.
- Extra Plots: Creation of complex plots combining default ones based on the already existing variables or the new ones that are declared in the New Calculations table.
- History: Evolution of parameters over a long time span.
- Expert Settings: The user can add expert code in order to include more functionalities than we already provide.

Different analysis management options are already available. Through these options the user can deploy the configurations to the production server, import existing configurations, compare if there is a difference between the current modifications and the production server and finally re-run the analysis. A new user can also create an analysis for an instrument by initially adding an empty one or importing an existing one. An overview table with the current status of all the analyses gives details about the production state, including inconsistencies between development and production configuration.

Technologies Involved

The Webapp is running in the local network of CERN with limited access to the CERN users only with a Single-Sign-On (SSO). For the development we use Django [4], an open source web application framework written in Python. The data analysis code is written in Python using libraries such as Plotly, NumPy, SciPy, Pandas. Using Gitlab's CI/CD tool the Webapp is deployed on Openshift after every change in the code, once the unit test have passed.

EXAMPLE USE CASES

We have selected two use cases to illustrate where the OAF is being used today. A full list would be too long to be covered in the paper. At the end of the LHC run 2 (end of 2018) there were nearly 100 use cases running daily.

The ability to measure luminosity on an absolute scale is essential for CERN's Large Hadron Collider (LHC) physics experiments [5]. The dominating contribution to the final uncertainty of the absolute luminosity calibration originates from the bunch current normalization. Thus, one important use case of OAF has been to assess and monitor our current different measurement performance. We are systematically cross-checking the measurements of the four DC BCTs (Beam Current Transformer) and the two fast BCTs to detect deviations between them. The calibration of all these instruments is compared with lab measurements to make sure that the absolute values are correct.

A second simpler example is the regular surveillance of the control of the water cooled racks temperatures hosting our Beam Position measurement electronics to prevent calibration changes caused by the building temperature variations (Fig. 8). We have to make sure that the temperature variations are below 1°C so that the orbit calculations are not affected.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV042

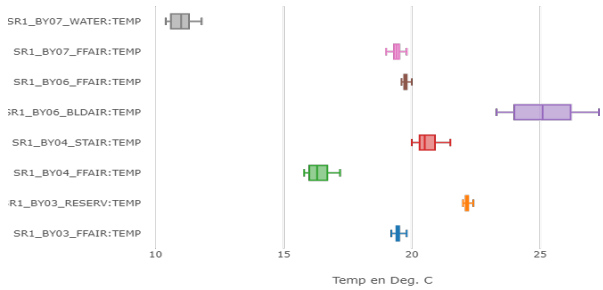


Figure 8: Temperature evolution envelop (min, max, median and a box from 1st quartile to 3rd) of the different sensors located in LHC building SR1.

EVOLUTION

We have several ideas to improve OAF in the short term. For the whole life cycle of an analysis, it is still necessary to open different tools. Our goal is to have a single entry point application that will allow to configure all the required steps (data logging, data preprocessing, data queries, analysis configuration, visualization, export).

We are also looking to have a full Python code base. Today we still need some Java functions to query the Spark based Accelerator Logging (NXCALS).

In the medium term, we want to improve and modernize the analysis capabilities of OAF to be able to cover new use cases. We are currently surveying the needs of users in the Beam Instrumentation group.

CONCLUSION

The Offline Analysis Framework was developed some years ago to systematically analyze beam instrumentation data. The tool has helped diagnose and solve numerous issues so far. In the meantime, the technologies it is based on have evolved and opened new possibilities. This paper presents a first step towards this direction by introducing a web-based self-service tool for the users to configure their analysis and visualize the results. A second future step will be to focus on the extensions of the analyses algorithms that can be used within the tool.

ACKNOWLEDGEMENT

We would like to thank our colleagues in the hardware teams of the Beam Instrumentation group for their input of new features needed for OAF. We would also like to thank our colleagues in the Controls Group for providing the infrastructure that OAF requires to run.

REFERENCES

[1] S. Jackson, C. Roderick, and C. Zamantzas, “A Framework for Off-line Verification of Beam Instrumentation Systems at CERN”, in Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13), San Francisco, CA, USA, Oct. 2013, paper MOPPC139, pp. 435-438.

[2] B. Kolad, J-J. Gras, S. Jackson, and S. B. Pedersen, “The CERN Beam Instrumentation Group Offline Analysis Framework”, in Proc. 5th Int. Beam Instrumentation Conf. (IBIC'16), Barcelona, Spain, Sep. 2016, pp. 449-452. doi:10.18429/JACoW-IBIC2016-TUPG45

[3] J. P. Wozniak and C. Roderick, “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”, presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA163.

[4] django - The web framework for perfectionists with deadlines, https://www.djangoproject.com/

[5] G. Anders et al., “LHC bunch current normalisation for the April-May 2010 luminosity calibration measurements.”, February ‘11, https://cds.cern.ch/record/1325370

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV043

USING AI FOR MANAGEMENT OF FIELD EMISSION IN SRF LINACS

A. Carpenter†, P. Degtiarenko, R. Suleiman, C. Tennant, D. Turner, L. S. Vidyaratne, Jefferson Lab, Newport News, Virginia, USA
K. Iftekharuddin, Md. Monibor Rahman, ODU Vision Lab, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, Virginia, USA

Abstract

Field emission control, mitigation, and reduction is critical for reliable operation of high gradient superconducting radio-frequency (SRF) accelerators. With the SRF cavities at high gradients, the field emission of electrons from cavity walls can occur and will impact the operational gradient, radiological environment via activated components, and reliability of CEBAF's two linacs. A new effort has started to minimize field emission in the CEBAF linacs by re-distributing cavity gradients. To measure radiation levels, newly designed neutron and gamma radiation dose rate monitors have been installed in both linacs. Artificial intelligence (AI) techniques will be used to identify cavities with high levels of field emission based on control system data such as radiation levels, cryogenic readbacks, and vacuum loads. The gradients on the most offending cavities will be reduced and compensated for by increasing the gradients on least offensive cavities. Training data will be collected during this year's operational program and initial implementation of AI models will be deployed. Preliminary results and future plans are presented.

INTRODUCTION

The Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab is a high power, continuous wave recirculating linac that completed an energy enhancing upgrade to 12 GeV in 2017 [1]. This upgrade included the installation of 11 additional higher gradient cryomodules, named C100s for their capability of producing a 100 MeV energy gain. Field emission (FE) is a well-known phenomenon in superconducting radio-frequency (SRF) cavities that can have deleterious impact on accelerator hardware, cryogenic heat loads, and machine operations. Field emitted electrons can be accelerated similarly to CEBAF's electron beam and can generate neutron and gamma radiation on impact. Managing FE in CEBAF's C100 cryomodules has emerged as an on-going operational challenge since the 12 GeV upgrade (Fig. 1).

CEBAF recently designed, built, calibrated, and installed neutron dose rate meters (NDX) [2]. The NDX monitors are deployed around CEBAF with a majority of detectors placed near the newer higher gradient cryomodules. This new system allows for more detailed measurements to be made of the radiation response to RF configurations and is currently being used to minimize the FE-based radiation through manual gradient optimizations.

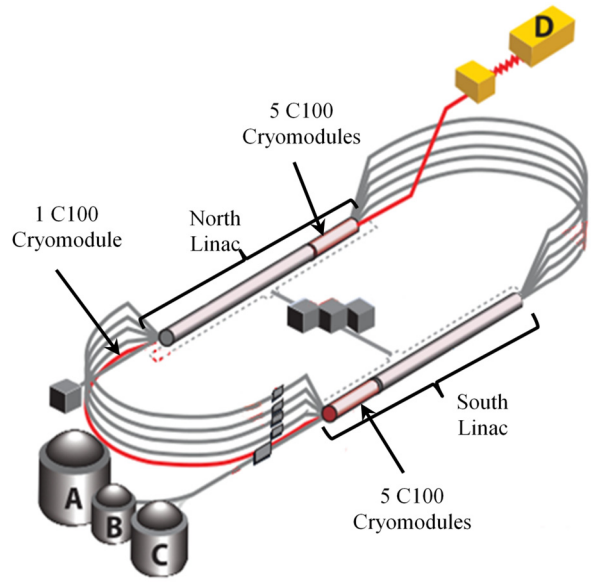


Figure 1: CEBAF schematic denoting the location of C100 cryomodules. One north linac C100 was removed for refurbishment during the time of this study.

Several beam studies were conducted during CEBAF restoration that leveraged the NDX system to measure FE-related radiation response to changes in cavity RF gradients. This data provides an ample training set for the development of artificial intelligence (AI) models to aid operations in maintaining a lower radiation environment. Preliminary attempts at modeling radiation as a function of gradient appear successful.

NDX SYSTEM

Installation and commissioning of the NDX system was completed in August 2021. The system has 21 detectors positioned at strategic locations in the CEBAF tunnel. The majority of these detectors are positioned around the newer higher gradient cryomodules with names corresponding to the adjacent downstream cryomodule. These detectors are primarily designed to measure neutron radiation, but as an ancillary and necessary feature, they also provide measurements of gamma radiation dose rates. The NDX system is now the primary tool for measuring FE-related radiation at CEBAF.

Electrometers associated with the detectors measure the current signal over a variable integration time period, typically set to one second. These signals are converted to

* This work is supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Contract No. DE-AC05-06OR23177.
†adamc@jlab.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV043

dose rates, presented over CEBAF's standard EPICS control system, and stored in the control system archiver.

PROBLEM OF FIELD EMISSION

SRF cavity walls may emit electrons when exposed to a sufficiently high RF gradient, which are subsequently accelerated by the same RF field. Electrons are emitted with an exponential response to increases beyond this onset threshold. Many of these electrons quickly impact the cavity and are reabsorbed, potentially causing increased cryogenic heat load or radiation. However, some FE electrons are captured by RF fields and transported long distances (>100 m) through adjacent cavities and cryomodules either upstream or downstream [3]. These electrons eventually collide with accelerator hardware producing much higher radiation levels at the new location.

This radiation has several deleterious effects on CEBAF operations. For example, hardware can be damaged requiring early replacement, components can become activated which present hazards for nearby work, and the increased radiation and field emission can increase the number of machine trips or lower the maximum operational gradient ("operational drive high") that can be achieved. These issues all contribute to increased CEBAF downtime and lowered energy reach. Field emission control, reduction, and management is critical for reliable, high gradient operation at CEBAF.

Field emission during CEBAF operations is largely impacted by contamination on the cavity wall surface. SRF cavity fabrication uses state-of-the-art surface processing and assembly techniques to control FE, however, particulates may be introduced through activities such as vacuum valve operations or installation work [4, 5]. Trace gasses may freeze on to cavity wall surfaces and activate or degrade existing field emitters [5]. These gasses may also be removed during warm-up events or over time during RF operations.

Ideally, all cavities could be run below the FE onset gradient. However, CEBAF's experimental requirements demand that many cavities are set beyond this threshold in order to meet the target energy of experimenters. Thus the question becomes how to best distribute gradient in order to reduce FE across a linac. CEBAF operations staff already have tools available for automatic gradient distribution to optimize for common cavity faults and other operational criteria. Our goal is to first develop AI tools to help operators better leverage the existing toolkit, rather than replace the existing automated gradient distribution process.

During summer 2021, operations staff used manual investigations of FE-related radiation response to changes in gradient to achieve large-scale reductions in radiation while maintaining linac energy gain. Use of machine learning (ML) models and advanced optimization techniques may be able to surpass these manually found settings. Additionally, operational conditions evolve and change during an experimental run. Existing field emitters may degrade or new ones appear. Operational limits on cavity gradients change as hardware fails or is repaired. As such operations would need to continue fine tuning this optimization throughout a run as time and manpower allow. AI may be able to provide similar functionality without the need for time consuming manual efforts.

These operational characteristics raise interesting questions regarding the management of FE:

1. Given a machine configuration, can the cavities that are the leading contributors to FE-radiation be identified? This would allow for off-line optimization work to be performed without interrupting beam delivery.
2. Can changes in existing field emitters be detected and localized? This would allow degraded or improved field emitters to be identified for manual operator intervention or possibly to inform updates to existing ML model regarding the previous question.
3. Can the appearance or elimination of field emitters be detected and localized? Completely new field emitters would likely pose a challenge for ML models trained on old data. Quickly identifying these would alert users that the model needs to be re-trained, and improve the rapidity of manual interventions.

Currently, these questions can be answered manually by invasively adjusting cavity gradients to explore the machine response. However, this process can take hours. This level of effort and beam studies time is difficult to obtain during experimental runs. Providing non-invasive AI methods to replace or enhance existing manual optimizations could enable CEBAF to maintain a lower level of FE during operations.

DATA COLLECTION METHODS

Our initial approach to these problems is to focus on the response of the C100 cryomodules at the end of the north linac. Two types of data were collected during beam studies, the radiation onset for C100 cavities, and radiation responses to a range of operational C100 gradient settings. First, the radiation onsets of C100 cavities were determined under pseudo-operational conditions. This is subtly different, but closely related to, a cavity's FE onset. The radiation onset measurements determine the highest gradient a cavity can achieve before the NDX system can definitively detect an increase above background radiation in a configuration approximating normal beam operations. Secondly, we measured the radiation response across the linac using the NDX system while scanning a range of gradients consistent with normal operations.

Radiation Onset

Automated radiation onset measurements were performed via software on one cryomodule at a time. First we turn off RF in at least the adjacent four cryomodules on either side to remove radiation generated by other cavities. Then all cavity gradients in the cryomodule of interest were increased as much as possible without causing a noticeable rise in radiation.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV043

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Once this elevated baseline was achieved, each cavity was individually walked up in 0.125 MV/m steps until a significant increase in radiation was observed. Determining if radiation levels exceeded background was done using a statistical comparison. Unconverted NDX detector current signals were sampled for ten seconds at the start of the scan to establish the background. Then similar samples were taken after each step. A difference in average currents of ten standard errors was considered statistically significant. Our statistical approach typically corresponded to a dose rate increase on the order of 1-10 mrem/h. This automated onset scanning procedure requires approximately one hour to find the radiation onsets for the eight cavities in a cryomodule. However, we believe there is opportunity for considerable speed enhancements.

Gradient Scan

A gradient scan was conducted by setting the entire linac to RF settings consistent with an energy gain used during experimental runs. This process sets all C100 cavities to their operational maximum gradients (drive high limits). From this starting point, we systematically stepped all of the C100 cavities' gradients down.

Each stage of a gradient scan consisted of stepping down individual cavity gradients in identical step sizes and in a randomized order. After turning down a cavity, we waited several seconds for the cryogenic system to settle, then allowed several seconds for dedicated radiation measurements. Data was collected for both the settle and dedicated measurement periods as radiation responses appear to be similar during both phases.

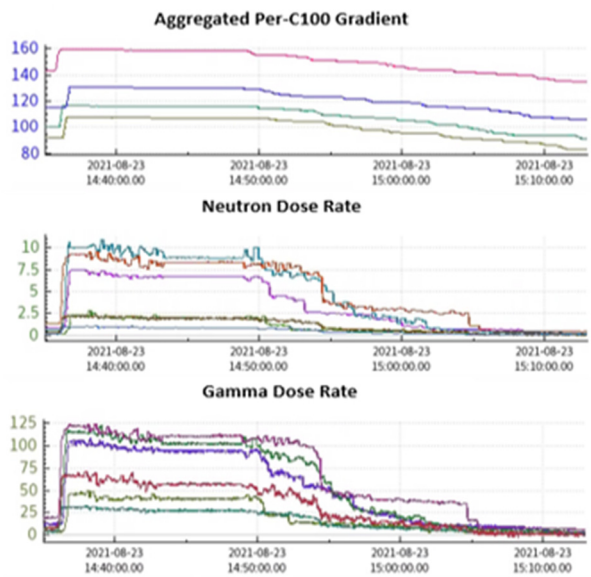


Figure 2: A single gradient scan using three 1 MV/m steps (top). The measured neutron (middle) and gamma (bottom) dose rates (rem/h) show the reduction in radiation as cavity gradients are lowered.

Figure 2 shows a single gradient scan and its radiation response. Notice that the reduction in radiation is a mix of plateaus and steep declines. This likely indicates that specific cavities were the primary field emitters as radiation levels dropped with small changes to gradient. In this example, radiation is practically eliminated while gradients have been indiscriminately reduced 15-20% below their standard operational settings. A more optimal approach could likely achieve similar radiation reductions while sparing much of the gradient losses.

A single gradient scan consisted of several such identical stages. A typical scan with three stages could be completed in approximately 20 minutes. As was the case with onset scans, there are opportunities to speed up the process by either collecting fewer samples per step, requiring less wait time after gradient change to collect data, or by algorithmic enhancements.

Multiple scans were performed at various step sizes ranging from 0.1 MV/m to 1 MV/m, and were performed starting at various offsets from the C100 cavities' maximum gradients. This allowed for a range of gradient combinations covering the highest 3 MV/m range of each cavity to be explored. All gradient scan data collection was managed by CEBAF's EPICS control system archiver, with the data collection software maintaining an index file for later retrieval.

DATA EXPLORATION

The gradient scans produced 17,940 samples (10 samples for each of the 1,794 gradient combinations explored during the scans). After data cleaning, 17,610 samples remained. The gradient scans produced a broad range in radiation mimicking dose rates that will be seen during operations (Fig. 3). Higher dose rates are possible but not achievable during beam studies without exceeding the current operational limits.

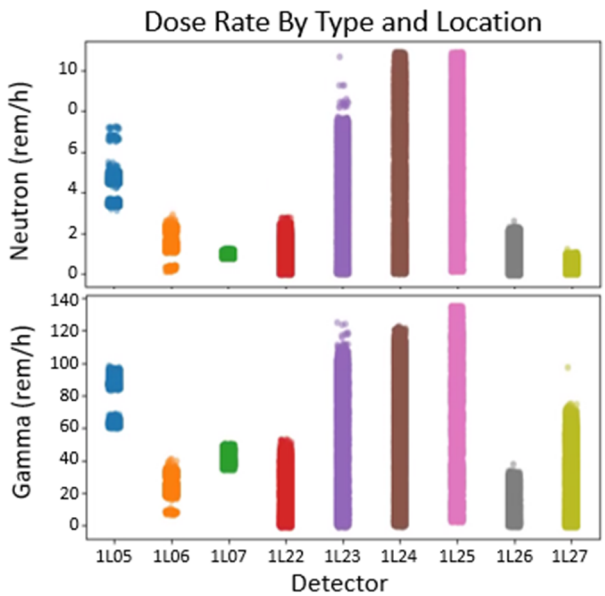


Figure 3: Radiation dose rate measured by NDX detectors during gradient scan studies. Each dot represents a single one-second integrated measurement. Six detectors (1L22-1L27) are positioned near C100s in the north linac.

C100 gradient settings were very positively correlated with individual radiation readings and radiation readings

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV043

were highly correlated amongst detectors near the C100 cryomodules (Fig. 4). Positive correlations were expected between gradients and detector readings given that, in general, higher gradients lead to higher radiation. Readings between detectors were also expected to be positively correlated as they occupy the same linac tunnel. However, it is possible that the structure of the gradient scans, where all C100 cavity gradients were stepped down in stages, exaggerated the correlations between detectors. Future gradients scans should keep this concern under consideration and allow additional randomness in the scanning procedure.

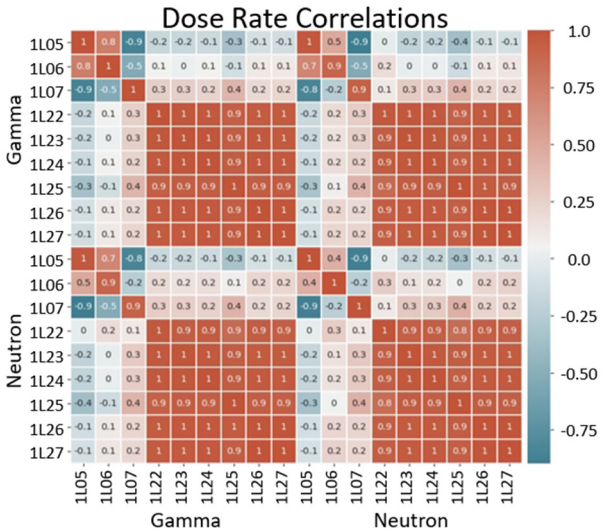


Figure 4: Dose rates were very positively correlated among the C100-adjacent NDX detectors. Correlations between other detectors were likely due to changes in non-C100 configurations between beam studies. Note that the same cryomodules are listed twice, once for gamma radiation and once for neutron.

MODELING RESULTS

We developed a preliminary model to address the first operational FE problem, i.e., identifying cavities that are leading offenders. Our initial approach is to directly model the radiation produced at all C100-adjacent detectors as a function of C100 cavity gradients and radiation onset values. Given a sufficient model, standard "black box" optimization techniques can be used to optimize the gradient settings, or an automated procedure can check which gradients will have the most anticipated impact on radiation production.

Preliminary attempts at modeling the radiation readings as a function of cavity gradients and radiation onset values used a multi-output random forest regressor [6] trained on 12,302 examples and tested on 5,308 examples for a 70/30 split. Examples taken from repeated measurements of a gradient configuration were grouped exclusively into the training or testing set to ensure the test data was unseen during training. Model development was performed using the scikit-learn python package [7].

Early attempts to model radiation using solely untransformed cavity gradients were unsuccessful. However, providing a feature set that more closely mimics the described physical process proved effective [3]. For this model, the following five features per cavity were engineered:

1. Surface FE: $g_i = 2^{max(gmes_i - rad_onset_i,\ 0)}$
2. Upstream energy gain: $u_i = \sum_j (gmes_j)$ where cavity j is upstream of cavity i
3. Downstream energy gain: $d_i = \sum_j (gmes_j)$ where cavity j is downstream of cavity i
4. Upstream interactions: $u_i g_i$
5. Downstream interaction: $d_i g_i$

Where $gmes_i$ is the measured cavity gradient of cavity i, and rad_onset_i is the radiation onset gradient of cavity i found during beam studies. These features, while crude approximations of the physical processes, are sufficient as the model achieved an R-Squared score of 0.978 using these features. Figure 5 shows the difference in observed and predicted neutron dose rates detected upstream of 1L25 as the aggregate C100 gradient changed. Table 1 gives additional metrics for model performance.

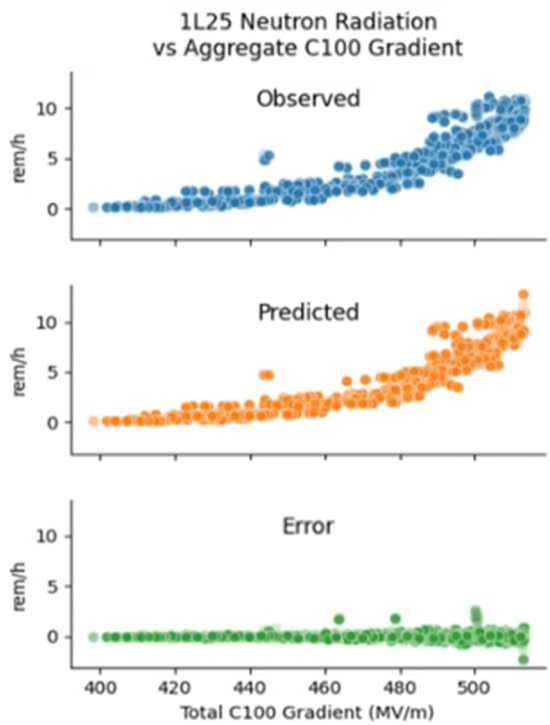


Figure 5: Testing results of the random forest model predicting neutron dose rates at detector 1L25. The model maintains small errors across the range of gradients.

A significant drawback of this approach is that the model will need to be retrained on new data when there are changes to active field emitters (i.e. if they are processed away or their onset changes). Additional work will investigate alternative modeling approaches or mitigations of this shortcoming, such as developing procedures for rapid or non-invasive data collection.

The initial modeling results provide confidence that the data from the NDX detectors, along with machine learning

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV043

techniques, can be leveraged to address the three operational questions highlighted earlier.

Table 1: Performance Metrics of the Multi-Output Random Forest Regressor

Metric	Training	Testing
R-Squared	0.999	0.978
MSE	0.001	0.052
MAE	0.013	0.115

SUMMARY

We have begun leveraging the NDX system to develop AI models capable of enhancing the management of field emission during CEBAF operations. Initial data collection and modeling efforts show promising signs that this is an effective tool to apply for that purpose. Our future work aims to refine this work with more advanced deep learning techniques and to expand the scope of the problems investigated.

REFERENCES

[1] C. E. Reece, "Continuous wave superconducting radio frequency electron linac for nuclear physics research", *Physical Review Accelerators and Beams*, vol. 19, no. 12, p. 124801, 12/28/ 2016. doi:10.1103/PhysRevAccelBeams.19.124801

[2] P. Degtiarenko, "Neutron Dose Rate Meters with Extended Capabilities", in *Proc. IEEE Nuclear Science Symposium (NSS) and Medical Imaging Conference (MIC)* in Manchester, UK, October-November 2019.

[3] R. Geng *et al.*, "Field Emission in Superconducting Accelerators: Instrumented Measurements for Its Understanding and Mitigation", in *Proc. 6th Int. Beam Instrumentation Conf. (IBIC'17)*, Grand Rapids, MI, USA, Aug. 2017, paper TH1AB1, pp. 470-477. doi:10.18429/JACoW-IBIC2017-TH1AB1

[4] A. Burrill *et al.*, "SRF Cavity Performance Overview for the 12 GeV Upgrade", in *Proc. 3rd Int. Particle Accelerator Conf. (IPAC'12)*, New Orleans, LA, USA, May 2012, paper WEPPC089, pp. 2423-2425.

[5] R. Geng, "Root Causes of Field Emitters in SRF Cavities Placed in CEBAF Tunnel", in *Proc. 7th Int. Particle Accelerator Conf. (IPAC'16)*, Busan, Korea, May 2016, paper THOBB03, pp. 3198-3201. doi:10.18429/JACoW-IPAC2016-THOBB03

[6] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, pp. 5–32, 2001. doi:10.1023/A:1010933404324

[7] scikit-learn, https://scikit-learn.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV046

VIRTUALIZED CONTROL SYSTEM INFRASTRUCTURE AT LINAC PROJECT PINSTECH

N. U. Saqib†, F. Sher*
LINAC Project, PINSTECH, Islamabad, Pakistan

Abstract

IT infrastructure is backbone of modern big science accelerator control systems. Accelerator Controls and Electronics (ACE) Group is responsible for controls, electronics and IT infrastructure for Medical and Industrial NDT (Non-Destructive Testing) linear accelerator prototypes at LINAC Project, PINSTECH. All of the control system components such as EPICS IOCs, Operator Interfaces, Databases and various servers are virtualized using VMware vSphere and VMware Horizon technologies. This paper describes the current IT design and development structure that is supporting the control systems of the linear accelerators efficiently and effectively.

INTRODUCTION

LINAC Project, PINSTECH aims at developing indigenous RF linear accelerators for medical and industrial purposes. Along with progressive research in this field, prototypes of 6 MeV Medical and Industrial (NDT) linear accelerators are being developed. IT infrastructure provides computing, network and storage resources, as well as several services to the accelerator control system, and also to engineers and scientists involved in research and development tasks. An Ethernet based Local Area Network (LAN) is deployed at LINAC Project. The overall LAN is divided into two segments: *Technical Network* and *Office Network*. Both networks are isolated and contain independent IT infrastructure. Measurement and control devices including commercial-off-the-shelf equipment and related devices are connected to the *Technical Network* while *Office Network* consists of users related devices including scientists' and engineers' PCs, printers, scanners, test equipment etc. For compute resources, Dell EMC PowerEdge servers are configured and installed to provide reliable services for both the networks. Servers are managed remotely via Integrated Dell Remote Access Controller (iDRAC) which eliminates the need of physical presence of server administrator to configure/reconfigure a server. For network resources, Allied Telesis Gigabit Ethernet network switches are installed to provide network connectivity and high bandwidth data transfer between nodes in the networks. Three types of network switches are utilized: unmanaged, web-smart and managed according to purpose of utilization. As multi-core processors become common, virtualization is an important technology to implement full utilization of hardware resources and reduce their footprint [1]. Virtualization enables running multiple virtual PCs simultaneously on one physical machine and allows multiple operating systems to work at the same time and on the same machine. Main advantage of virtual machines is that hardware resources such as processor, memory, hard disk, device controllers, network cards etc. can be dynamically increased or decreased according to the requirements.

SERVER VIRTUALIZATION

Server virtualization divides a physical server into multiple virtual machines by means of a software layer called hypervisor. Virtualization fully utilizes a physical server by distributing workload on virtual servers. Additional benefits include less hardware to buy and manage, more efficient resources usage and improved resilience. Dell EMC PowerEdge servers [2] are virtualized using VMware vSphere 6 [3] technology which is a data center virtualization solution. VMware vSphere consists of two components: VMware ESXi which is a type 1 hypervisor installed on bare metal server and vCenter Server which provides centralized management platform for ESXi hosts, virtual machines and other dependent components. Two Dell R740 servers are deployed in the *Office Network* and one Dell R630 server is deployed in the *Technical Network*. Specifications of both types of servers are provided in Table 1 and Table 2 respectively.

Table 1: Specifications of Dell R740 Server

Component	Specification
CPU	Intel Xeon Silver @ 2.1 GHz x 2
RAM	64 GB (32 GB x 2)
HDD	8 TB (2 TB x 4)
OS	VMware ESXi 6.7

Table 2: Specifications of Dell R630 Server

Component	Specification
CPU	Intel Xeon Silver @ 2.1 GHz
RAM	32 GB (16 GB x 2)
HDD	6 TB (2 TB x 3)
OS	VMware ESXi 6.7

TECHNICAL NETWORK

All components related to control system consisting of servers, desktop computers and commercial-off-the-shelf equipment are connected to the *Technical Network* to pro-

† najm.control@gmail.com
† falaksher@gmail.com

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV046

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

vide isolation, security and reliability [4]. Dell EMC PowerEdge R630 virtualized server deployed in the *Technical Network* and shown in Fig. 1 provides the following services:

EPICS IOCs

EPICS (Experimental Physics and Industrial Control System) is a software toolkit to develop distributed control systems[5]. EPICS IOCs (Input/Output Controllers) deployed in our control system are soft IOCs operating in open source CentOS 7 operating system. IOCs are suitable to be run in virtual machines because none of the devices are directly attached to the hardware. Also, serial interfaces are converted to Ethernet by means of MOXA and USR-IOT Serial-to-Ethernet converters wherever possible.

Operator Interfaces

Operator Interfaces (OPI) are developed in Phoebus Control System Studio (CSS) which is a set of tools and application for developing Graphical User Interfaces (GUI) [6]. CSS is installed in a virtual machine and OPIs are designed and developed on it. OPIs are then deployed to *Operator Computer* in the *Control Room*.

Data Archiver and Alarm Services

Data archiver for recording history of process variables is implemented using CSS Best Ever Archive Toolset, Yet (BEAUTY). Alarm services for notifications about events is implemented using CSS Best Ever Alarm System Toolkit (BEAST). Both of these services are configured in CentOS 7 OS based virtual machine.

Database

Relational database for data archiver system is provided via MySQL server running in Microsoft Windows virtual machine.

Network Services

Domain controller residing in a virtual machine provides essential network services such as Active Directory Domain Services (ADDS), Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), Windows Deployment Services (WDS) and File Transfer Protocol (FTP) servers.

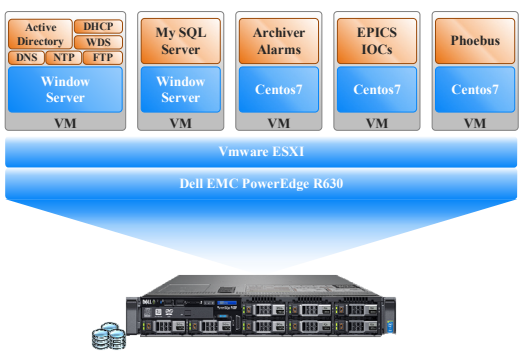


Figure 1: Technical Network Virtualization.

OFFICE NETWORK

General network consists of personal computers of engineers and scientists, workstations for simulations, servers for services and other equipment such as printers, scanners, test equipment etc. Two Dell EMC PowerEdge R740 servers deployed in the office network are virtualized using VMware ESXi 6.7 as shown in Fig. 2. These servers provide the following services:

vCenter Server

Open SUSE OS based vCenter Server appliance for managing ESXi hosts, virtual machines, network and storage.

Network Services

Domain controller provides essential network services such as ADDS, DNS, DHCP, WDS and FTP servers.

VMware Horizon Server

Microsoft Windows Server 2016 VM contains VMware Horizon View 7.11 Connection Server for providing Virtual Desktop Infrastructure (VDI). It provides a web based client interface to manage VDI. Adding/removing desktop pools, assignment of virtual machines to the pools and adding/removing of users' entitlements are few of the several important tasks managed through the client.

Nextcloud Hub

Centos 7 VM for Nextcloud Hub [7] provides collaboration platform for scientists and engineers. It provides many features such as file sharing, storage, announcements, chat, contacts, photos etc.

GitLab Server

CentOS 7 VM for GitLab server provides version control system and much more. It is a private server consisting of five users and is being tested currently. After testing, it will be integrate into the system soon as it provides essential services for software development.

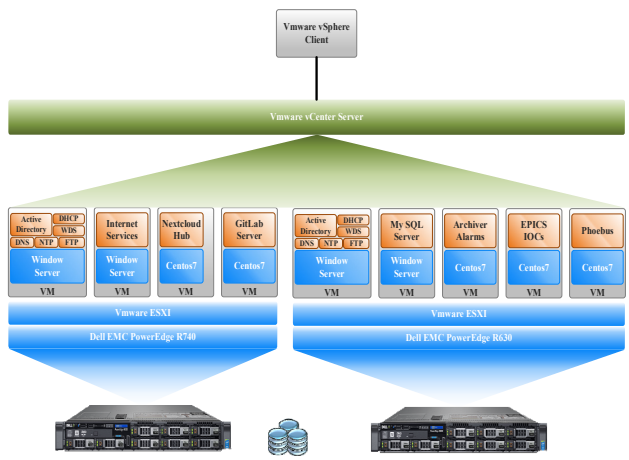


Figure 2: Office Network Virtualization.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV046

VIRTUAL DESKTOP INFRASTRUCTURE

Virtual Desktop Infrastructure refers to desktop hosting environment on a central server that manages and allows access of desktop and server operating systems running in virtualized machines, and deploys them to end-users on requests [8]. VMware Horizon 7 is used to provide desktop virtualization environment at our facility. Currently, main purpose of VDI is to provide internet access, and to provide remote workstations to scientists and engineers for helping out in their research and development tasks. Access to internet services is provided to users through a virtual machine that contains a dedicated network interface to a public network. HP Zero Clients and Horizon client software for Windows are used at end-user side to connect remotely to these desktops. Other desktops configured include dedicated user PCs for some users. Figure 3 presents layout of the deployed system.

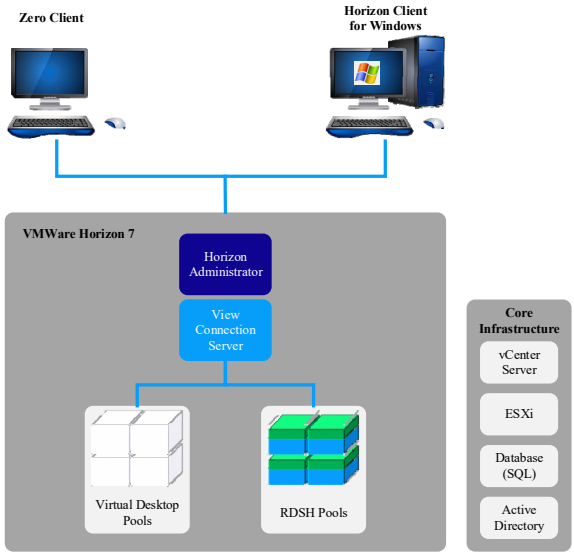


Figure 3: VDI Layout.

CONCLUSION

Virtualization technology has greatly improved development & deployment of the control system and services in terms of stability and reliability. Office Network is providing services as well as a collaboration platform for scientists and engineers. There is a room for a lot of improvement including new hardware and software technologies. We are gradually moving in the right direction but are limited with manpower and expertise. We expect significant upgrades in near future.

REFERENCES

[1] U. Felzmann, N. Hobbs, and A. C. Starritt, "Virtualisation within the Control System Environment at the Australian Synchrotron", in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, pp. 664-666. doi:10.18429/JACoW-ICALEPCS2015-WEM303

[2] Dell, https://www.dell.com

[3] VMware, https://www.vmware.com

[4] P. Kurdziel, "VDI (Virtual Desktop Infrastructure) Implementation for Control System - Overview and Analysis", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 501-502. doi:10.18429/JACoW-ICALEPCS2017-TUPHA048

[5] EPICS, https://www.epics-controls.org

[6] Control System Studio (CSS), https://www.controlsystemstudio.org

[7] Nextcloud, https://www.nextcloud.com

[8] S. W. Kim, H. J. Choi, H. S. Kim, and W. W. Lee, "A Virtualized Beamline Control and DAQ Environment at PAL", presented at the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WEPHA078, pp.1273-1275. doi:10.18429/JACoW-ICALEPCS2019-WEPHA078

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV047

STATUS OF HIGH LEVEL APPLICATION DEVELOPMENT FOR HEPS*

Xiaohan Lu†, Qiang Ye, Hongfei Ji, Yi Jiao, Jingyi Li, Cai Meng, Yuemei Peng, Gang Xu, Yaliang Zhao
Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China

Abstract

The High Energy Photon Source (HEPS) is a 6 GeV, 1.3 km, ultralow emittance ring-based light source in China. The construction started in 2019. In this year, the development of beam commissioning software of HEPS started. It was planned to use EPICS as the control system and Python as the main development tools for high level applications (HLAs). Python has very rich and mature modules to meet the challenging requirements of HEPS commissioning and operation, such as PyQt5 for graphical user interface (GUI) application development, PyEPICS and P4P for communicating with EPICS. A client-server framework was proposed for online calculations and always-running programs. Model based control is also one important design criteria, all the online commissioning software should be easily connected to a powerful virtual accelerator (VA) for comparison and predicting actual beam behaviour. It was planned to use elegant and Ocelot as the core calculation model of VA.

INTRODUCTION

The High Energy Photon Source (HEPS) is a 6 GeV, 1.3km, green-field 4th generation storage ring light source [1]. For this light source, the lattice design and related physics studies were basically finished [2-6], and the construction started in mid-2019. The machine commissioning of LINAC is expected to start in mid-2022. By this time, the high level applications (HLAs) for beam commissioning are indispensable.

To achieve this goal, the development of HLAs started this year. Before the start of development, we investigated the HLA schemes of almost all the running light sources. Most of them use the Matlab Middle Layer Toolkit (MML) [7, 8] as the main commissioning tools, and part of them developed new specific commissioning tools, such python-based aphla of NSLSII [9], and python-based HLA of Sirius [10]. Recently, python becomes more and more popular in every walk of life, due to its powerful modules and easy-to-learn characteristic. And more and more labs use python as the main development tools for control software and data analysis tool. In consideration of future development, economic and time cost, for the HEPS python was chosen as the main development tools, and EPICS as the control system. Python has very rich modules to meet the demands of HLA. PyQt5 was used to develop GUI app, will be update to pyqt6 in near future, PyEPICS and P4P were used as the communication tools. The HLA of HEPS is model-based, we decided to use Elegant [11] and Ocelot [12] as the core calculation models. A client-server framework was proposed for online calculations and always-running programs. All the online commissioning software should be easily connected to a powerful virtual accelerator (VA) for comparison and predicting actual beam behaviour.

APPLICATIONS FRAMEWORK

The development of HLA is collaborative work between the control group and physics group. The whole control system framework is shown in Fig. 1, all the applications were classified into three categories: parameter display and hardware controls, algorithm-based applications for tuning or measurements, model-based applications. The accelerator physics group is responsible for the development of the algorithm-based and model-based applications (the green part in Fig.1), and the control group is responsible for the development of the rest of applications and the construction of the low-level control applications (the blue part in Fig. 1).

The control group chose Control System Studio (CSS) [13] and python display management (PyDM) to build the parameter display and hardware control applications. The accelerator physics group is responsible to develop a brand new platform python accelerator physics application set (Pyapas) to build the HLA, as shown in Fig.2 with the aim of including all necessary tools and modules.

Pyapas is not only a set of applications, but also designed as a platform for the rapid development of HLA. Pyapas learns design philosophy from PyDM [14] and OpenXAL [15]. There are three principles of Pyapas, the first one is that all the applications are physical quantity based. The parameters we work with should be physical quantity. For example, when we do the orbit correction, we should change the magnetic field rather than the current of the correctors. The second one is that most of the applications should be server-based, the calculation process could be running in the background. The third one is that the applications should be model-based, all the applications could be connected to the VA, use the simulation results to tune and predict the behaviour of the beam.

According to the design principles, a clear roadmap was made to develop Pyapas. The drag-and-drop way should be implemented to develop GUI applications quickly. To develop server-based application, a robust and convenient client-server framework is indispensable. The model-based applications and VA require powerful physical models, for the moment we decide to use the exist mature simulation code Elegant and Ocelot as the core physical model. To

*Work supported by NSFC (12005239)
† luxh@ihep.ac.cn

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV047

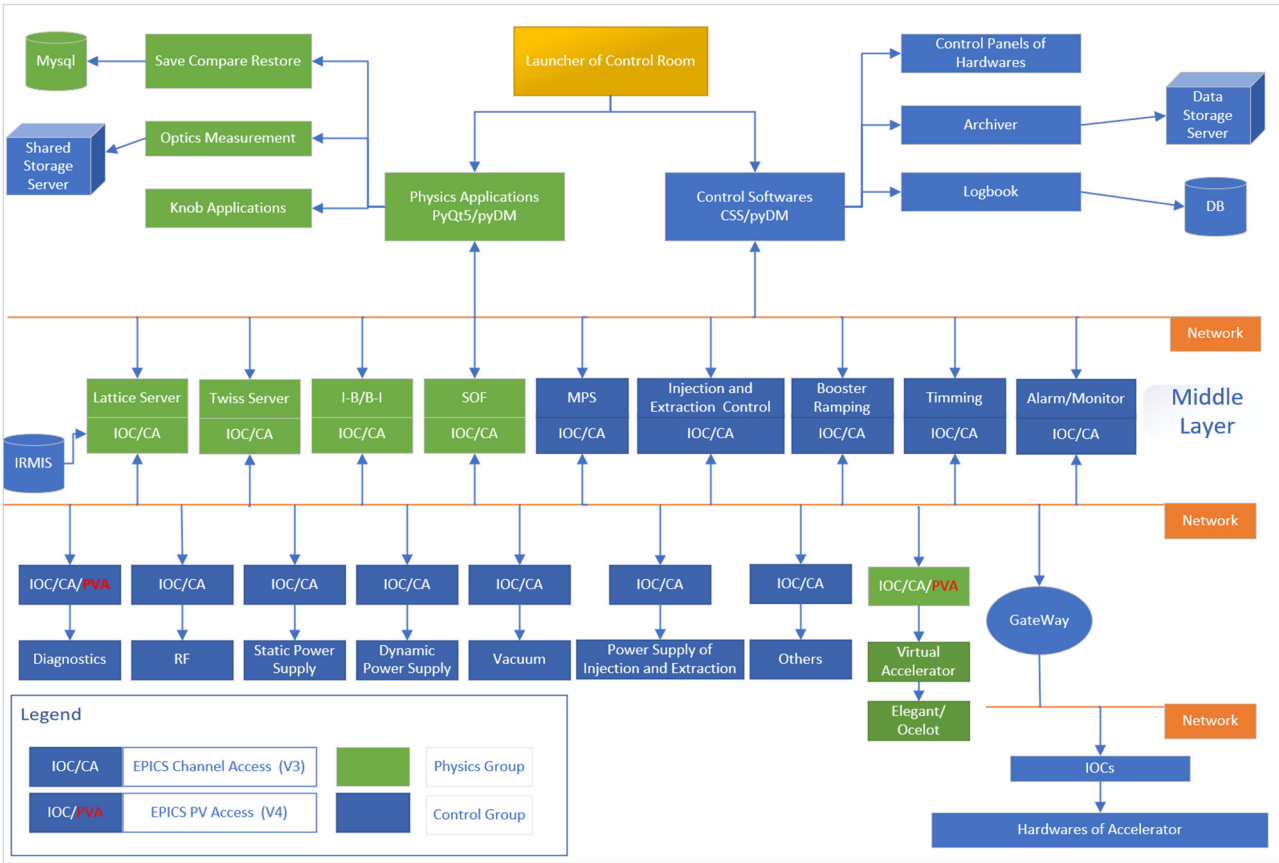


Figure 1: The structure of the control system.

make the development of model-based application intuitive, a new structure will be developed to describe the whole accelerator. For example there will be a magnet class to describe a real magnet, and an instance of this class will hold all the information of one magnet include channels and connector to call tracking algorithm. With these classes it is convenient to control machine and setup simulation, even control the machine based on simulation results. And the channel class based on EPICS for communication with machine should also be embedded. The adaptor for connecting to database is also on the development list.

The above development list describes the core elements of Pyapas, but it's not the ultimate goal. Machine learning is also on the development list, data acquisition and model training modules will be added to Pyapas. With trained model it will be easily to predict and control the actual machine through Pyapas.

To achieve the above goals, lots of development tasks have been started and some were already done. Extended QtDesigner was added to Pyapas as GUI development tool. Many customized widgets were embedded for rapid development, such as plot widgets, table widget with filter. The connectors were developed to connect to elegant and ocelot used as the core physical model. RPC and mDNS were implemented to build service applications, and adaptor for MySQL was also added to connect the database.

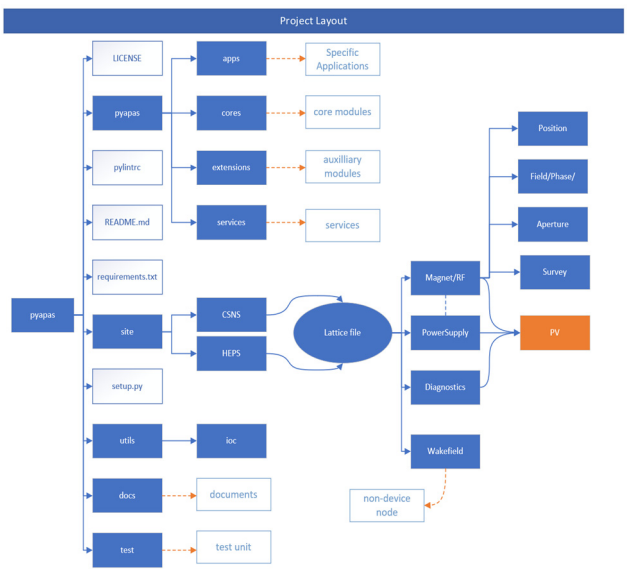


Figure 2: The structure of Pyapas.

APPLICATIONS UNDER DEVELOPMENT

For the requirements of the upcoming LINAC commissioning, the necessary applications are already under development, includes launcher, orbit correction, phase scan, emittance measurement, beam based alignment, energy and energy spread measurement, virtual accelerator and so on.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

As shown in Fig. 3, the launcher application is service-based. All the applications of Pyapas will register itself on the local network through mDNS, and the launcher could monitor the status of all the launched applications and do the close and launch actions.

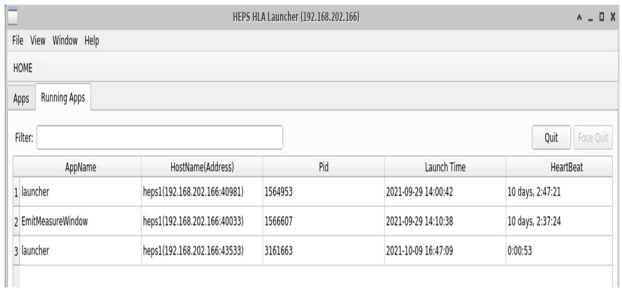


Figure 3: The snapshot of launcher.

The virtual accelerator shown in Fig. 4 is also service-based. There is a virtual IOC system which hold the same records of the real machine but start with prefix 'V'. The calculation engine built based on elegant and ocelot will supply the simulation values to the virtual IOC. For the commissioning software, the VA is same as the real machine, together with some auxiliary modules, we could also get the Twiss parameters in real time.

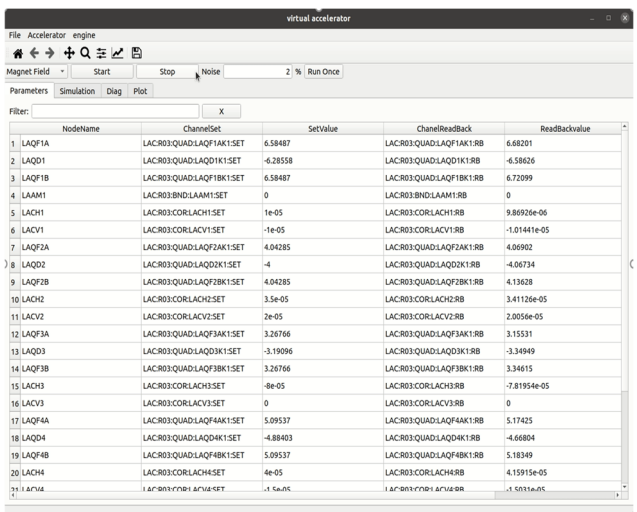


Figure 4: The snapshot of virtual accelerator.

CONCLUSION

The development of HLAs for HPES started this year. A new python-based flexible framework Pyapas is proposed to build HLA of HEPS. Pyapas is positioned as a platform for rapid development of beam commissioning software, it will be as convenient as CSS. Different from CSS for building display and hardware control applications, Pyapas is designed to build model-based applications. We keep adding new features and modules to it to make it more powerful.

To meet the requirements of upcoming beam commissioning of LINAC, the necessary applications are under development.

REFERENCES

[1] Y. Jiao, G. Xu, X. H. Cui *et al.*, "The HEPS project," J. Synchrotron Radiat. 25, 1611(2018).

[2] Y. Jiao *et al.*, "Modification and optimization of the storage ring lattice of the High Energy Photon Source", Radiat. Detect. Technol. Methods 4, 415–424, 2020. doi:10.1007/s41605-020-00189-7

[3] C. Meng *et al.*, "Physics design of the HEPS LINAC", Radiat. Detect. Technol. Methods 4, 497–506, 2020. doi:10.1007/s41605-020-00205-w

[4] Y.M. Peng *et al.*, "Design of the HEPS booster lattice", Radiat. Detect. Technol. Methods 4, 425–432, 2020. doi:10.1007/s41605-020-00202-z

[5] Y.Y. Guo *et al.*, "The transfer line design for the HEPS project", Radiat. Detect. Technol. Methods 4, 440–447, 2020. doi:10.1007/s41605-020-00209-6

[6] Y. Jiao *et al.*, "Progress of lattice design and physics studies on the High Energy Photon Source", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 229-232. doi: 10.18429/JACoW-IPAC2021-MOPAB053

[7] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LSAP- 302, 2005.

[8] J. Corbett, A. Terebilo, G. Portmann, "Accelerator Control Middle Layer," in *Proc. PAC 2003*, Portland, OR, USA, May 2003, pp. 2369-2371.

[9] L. Yang *et al.*, "The Design of NSLS-II High Level Physics Applications,", in *Proc. ICALEPCS2013*, San Francisco, CA, USA, Oct. 2013, paper TUPPC130, pp. 890-892.

[10] I. Stevani *et al.*, "High Level Applications for Sirius," in *Proc. PCaPAC2016*, Campinas, Brazil, Oct. 2016, paper WEPOPRPO22, pp. 47-49. doi:10.18429/JACoW-PCaPAC2016-WEPOPRPO22

[11] M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation", APS LS-287, 2000.

[12] I. Agapov *et al.*, "OCELOT: a software framework for synchrotron light source and FEL studies", Nucl. Instr. Meth. A, vol. 768, p. 151, 2014.

[13] Control System Studio, cs-studio.sourceforge.net

[14] PyDM - Python Display Manager, https://slaclab.github.io/pydm/

[15] A. P. Zhukov *et al.*, "Open XAL Status report 2019", in *Proc. IPAC2019*, Melbourne, Australia, May 2019, paper WEPTS096, pp. 3341-3344. doi: 10.18429/JACoW-IPAC2019-WEPTS096

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV048

NOVEL CONTROL SYSTEM FOR THE LHCb SCINTILLATING FIBRE TRACKER DETECTOR INFRASTRUCTURE

M. Ostrega, M. Ciupinski, S. Jakobsen, X. Pons
CERN CH-1211 Geneva 23, Switzerland

Abstract

During the Long Shutdown 2 of the LHC at CERN, the LHCb detector is upgraded to cope with higher instantaneous luminosities. The largest of the new trackers is based on the scintillating fibres (SciFi) read out by SIlicon PhotoMultipliers (SiPMs). The SiPMs will be cooled down to -40°C to minimize noise. For performance and integration compatibility, the cooling lines are vacuum insulated. Ionizing radiation requires detaching and displacing the readout electronics from Pirani gauges to a lower radiation area. To avoid condensation inside the SiPM boxes, the atmosphere inside must have a dew point of at most -45°C. The low dew point will be achieved by flushing a dry gas through the box. 576 flowmeters devices will be installed to monitor the gas flow continuously. A Condensation Prevention System (CPS) has been implemented as condensation was observed in previous detector operation tests. The CPS powers heating wires installed around the SiPM boxes and the vacuum bellows isolating the cooling lines. The CPS also includes 672 temperature sensors to monitor that all parts are warmer than the cavern dew point. The temperature readout systems are based on radiation tolerant multiplexing technology at the the front-end and a PLC in the back-end.

INTRODUCTION

SciFi Tracker

The SciFi tracker consists of three stations each with four detection planes. The detector is built from individual modules (0.5 m ×4.8 m), each comprising 8 fibre mats with a length of 2.4 m as active detector material. The fibre mats consist of 6 layers of densely packed blue-emitting scintillating fibres with a diameter of 250 µm (see Fig. 1). The scintillation light is recorded with arrays of state-of-the-art multi-channel silicon photomultipliers (SiPMs). A custom ASIC is used to digitize the SiPM signals. Subsequent digital electronics performs clustering and data-compression before the data is sent via optical links to the DAQ system. To reduce the thermal noise of the SiPM, in particular after being exposed to a neutron fluence of up to 10^{12} neq/cm^2, expected for the lifetime of the detector, the SiPM arrays are mounted in so called cold-boxes and cooled down by 3D-printed titanium cold-bars to −40˚C. The detector is designed to provide low material budget (1 % per layer), hit efficiency of 99 % and a resolution better than 100 µm. These performance figures must be maintained over the lifetime of the detector which will receive radiation dose up to 35 kGy near the beam pipe. The full detector, comprising 590 000 channels, is read out at 40 MHz [1].

Detector Infrastructure

Three control systems have been developed in order to allow the operation with correct environmental parameters for the SciFi infrastructure were developed. In this paper, the vacuum system, condensation protection system and flowcells monitoring system were described.

Figure 1: LHCb SciFi Tracker assembly.

VACUUM SYSTEM

System Overview

The SciFi vacuum system consists of two subsystems that serve two detector sides (A & C). Each subsystem consists of a scroll pump (primary) and two turbomolecular pumps connected parallel for redundancy purposes. “Turbo” pumps set is connected to the central manifold, where are the manual valves. Those valves can insulate every C-Frame in case of maintenance. There are 48 vacuum lines (12 supply and 12 return lines per side), connecting two main manifolds with the 12 detector C-frames.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV048

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Figure 2: Turbomolecular pumps shielding.

Both vacuum subsystems are connected by the primary vacuum bypass valve, which allows the whole system for operation in case of one primary pump failure. In such a situation, both turbomolecular pumps (side A & C) runs with the same primary pump. This increases the whole system's redundancy (two turbomolecular pumps per side and two primary pumps).

Turbomolecular pumps are located on both sides, in close proximityof the detector and required to be protected against magnetic field. To allow safe operation, pumps, electro valves and two pressure gauges have been placed in the 2-layer iron shielded case (see Fig. 2). Additionally, pump drivers have been detached and located in the PLC (Programmable Logic Controller) rack in the LHCb service cavern.

Vacuum System Controls

Each of the 12 C-Frames has two vacuum gauges for the upper manifold and lower manifold, 24 Pirani sensors in total. The presence of ionising radiation, cumulating to 50 Gy over the lifetime of the experiment, requires detaching the readout electronics from those Pirani gauges that are mounted on the manifolds (see Fig. 3). Two 3U racks (one per side) with 12 readout electronic PCB are located in the safe distance from the detector (in the bunker area). Tests shows that detaching electronic and sensor adds small offset on the readout value, which is acceptable.

The SciFi Vacuum Control System is based on Siemens S7-1500 PLC technology. It operate the system in the automatic and safe mode as well it assure the readings of the Pirani probes. The PLC program follows the UNICOS framework which allows the PLC program software connection to the WinCC OA SCADA software used in the Supervisor Layer [2][3].

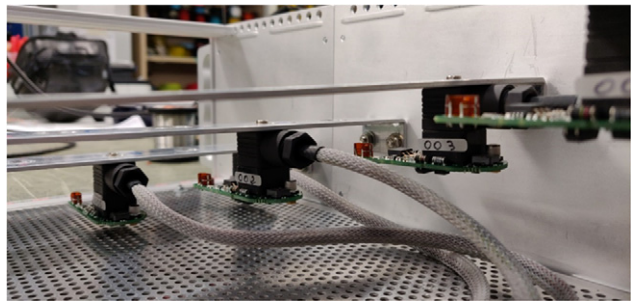


Figure 3: Pirani gauge electronic.

CONDENSATION PREVENTION SYSTEM

System Overview

Even though the good operation of the vacuum system, the collaboration has decided the implementation of high complex temperature measurement and heating system to avoid the condensation that may endanger the correct operation of the detector. This system, called CPS (Condensation Prevention System) consists in the installation of high-density temperature measurement network in the strategic places in the detector.

The CPS allows the monitoring, and the survey of 672 temperature sensors, Pt100 type, that are connected to radiation tolerant electronic interfaces, multiplexed and connected to a PLC, which collects all the information that is sent, a posteriori, to the Detector Control System. In case of condensation a heating wire that has been wrapped around the cooling transfer lines and the box housing the SiPMs. The heating wires can be powered by means of external power supplies. The heating power is distributed by the same electronic interfaces used for temperature measurements, in addition all the power consumption values of each circuit is measured by same interface through the PLC and sent up to the DCS for its control.

Each C-frame module (24 per C-frame) is equipped with one electrical heater and two Pt100 sensors. On each C-frame quadrant (4 per C-frame) there is one multiplexer box, which is responsible for reading 12 Pt100 signals and the measurement of the current of 6 electrical heaters. Additionally, on the MUX board, there is a power distribution and electrical protection (fuses) for six electrical heaters. One of the MUX installed on the C-frame, called MUX Master, will consist also additional multiplexer to multiplex all C-frame signals (see Fig. 4). Power to each heaters group is supplied by the MARATON power supplies. All the temperature signals (48 per C-frame, 576 in total) and heater power signals (24 per C-frame, 288 in total) are connected to the PLC located in the service cavern. Additionally, CPS monitor also cooling water temperature, which at the end gives total amount of 672 signals. All the MUX boards are based on electronics with sufficient radiation tolerance to cope with the radiation expected in the live time of the detector.

CPS Controls

The CPS control system is based on Siemens S7-1500 PLC technology commonly used and supported at CERN.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV048

The PLC control program follow the CERN BE-ICS control system standards (UNICOS-COMM) which allows the PLC program software connection to the WinCC OA SCADA software used in the Supervision Layer (i.e. the SciFi Detector Control System or DCS). The WinCC OA software also archive all the relevant data in an ORACLE database.

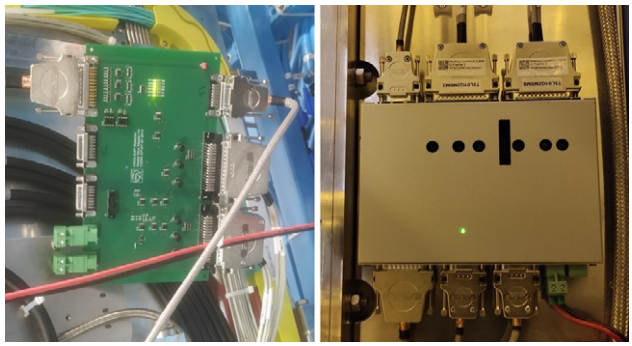


Figure 4. CPS Master multiplexing board.

FLOWCELLS DAQ SYSTEM

System Overview

The SiPM photodetectors are attached to cold bars through which Novec at about -40°C is flowing. The low temperature operation of the SiPMs mitigates radiation-related dark noise. To avoid condensation and frost formation inside the cold box, in particular on the SiPM arrays, the atmosphere inside the box must be free from humidity down to a dew point (DP) of approximately -50°C.

A complete sealing of the cold boxes is practically impossible, because the boxes need to ensure direct optical contact of the SiPMs to the scintillating fibres and also the passage of the SiPM signals via kapton flex cables to the electronic cards.

The low dew point will therefore be achieved by flushing a dry gas through the box. A small overpressure will compensate for potential leaks which could let humid air diffuse into the box. Flowmeter devices are installed on the outgoing line of each cold box in order to monitor continuously the gas flow. To insure a reliable measurement a fully redundant flow measurement is made with two flowmeters by outgoing line.

Controls

All the flow cells signals (48 per C-frame, 576 in total) are connected to two patch panels boxes, where power is distributed (two independent power supplies) and signals are separated (normal & redundant). The patch panel is connected with the DAQ rack using 12 MCA50 cables with the length around 60m each. The readout system is based on a 12 multiplexing boards (see Fig. 5). To preserve the redundancy, flow signals are separated into two multiplexers sub-racks. Multiplexed signals are connected to the SIEMENS S7-1500 PLC.

The PLC control program follows the CERN control system standards (UNICOS) which allows the PLC program software connection to the WinCC OA SCADA software used in the Supervision Layer. The WinCC OA software will also archive all the relevant data in an ORACLE database.

The PLC is installed in a dedicated 19'' rack for the dry gas DAQ system, which also contain the two multiplexers crate and two DC power supplies. Rack is also shared with the CPS PLC.

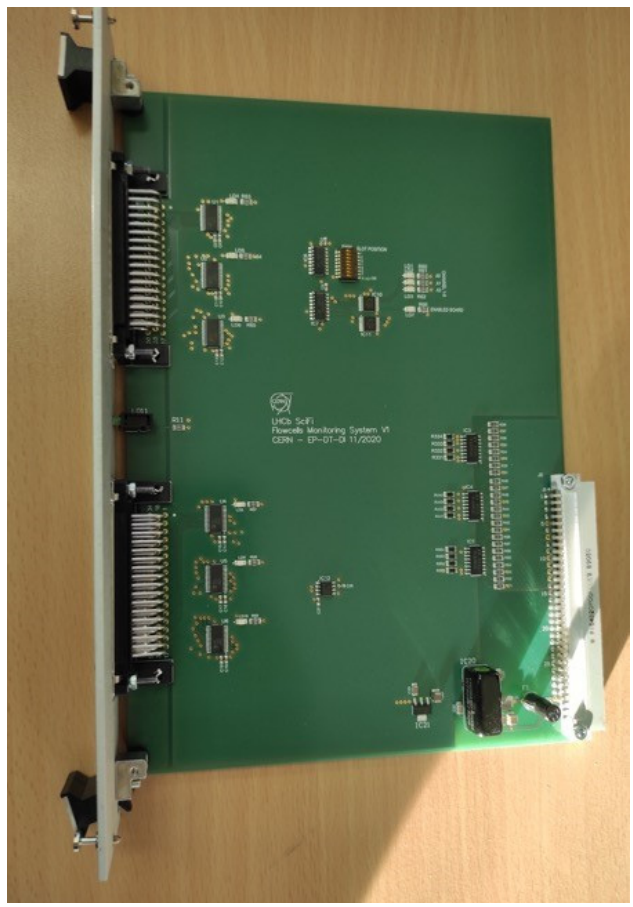


Figure 5. Flowcells multiplexing circuit.

SUMMARY

The multiplexing technology, used in the CPS and flowcells systems might be an interesting solution for the slow controls or monitoring systems. In the SciFi CPS system, almost 1000 channels are multiplexed and read by the PLC which is equipped with one DO (Digital Output) card and only three AI (Analog Input) cards. The multiplexers are installed on the detector which significantly reduces the number of cables between the experimental and service cavern, where the PLC is located. In the flowcells system, 12 multiplexers boards are installed in the accessible area, 576 signals are connected to the three AI cards, which reduce the signal readout time. The use of the Pirani gauges, with the detached electronic, reduced the cost of the vacuum system. Tests show that detaching the electronic has an acceptable effect on the readout vacuum pressure. It

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV048

might be also an interesting solution for the systems with a high number of readouts points, where vacuum sensors are affected by radiation.

REFERENCES

[1] P. Hopchev, "SciFi: A large Scintillating Fibre Tracker for LHCb", The Fifth Annual Conference on Large Hadron Collider Physics, Shanghai, China, May 2017.

[2] Ph. Gayet *et al.*, "UNICOS a Framework to Build Industry-like Control Systems Principles Methodo-logy", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper WE2.2-6I.

[3] B. Fernandez Adiego *et al.*, "UNICOS CPC6: automated code generation for process control applications", in *Proc. ICALEPCS 2011*, Grenoble, France, Oct. 2011, paper WEPKS033, pp. 871-874.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV049

VIRTUALISATION AND SOFTWARE APPLIANCES AS MEANS FOR DEPLOYMENT OF SCADA IN ISOLATED SYSTEMS

P. Golonka†, L. Davoine, M. Zimny, L. Zwalinski, CERN, Geneva, Switzerland

Abstract

The paper discusses the use of virtualisation as a way to deliver a complete pre-configured SCADA (Supervisory Control And Data Acquisition) application as a software appliance to ease its deployment and maintenance. For the off-premise control systems, it allows for deployment to be performed by the local IT servicing teams with no particular control-specific knowledge, providing a "turnkey" solution. The virtualisation of a complete desktop allows to deliver and reuse the existing feature-rich Human-Machine Interface experience for local operation; it also resolves the issues of hardware and software compatibilities in the deployment sites. The approach presented here was employed to provide replicas of the "LUCASZ" cooling system to collaborating laboratories, where the on-site knowledge of underlying technologies was not available and required to encapsulate the controls asa "black-box" so that for users, the system is operational soon after power is applied. The approach is generally applicable for international collaborations where control systems are contributed and need to be maintained by remote teams.

MOTIVATION

For the past two decades industrial controls technologies have been applied with success at CERN to build control systems for a vast range of applications, ranging from laboratory setups through technical infrastructure up to the ones for accelerators and large particle detectors counting millions of I/O channels. Applying the standardized stack of industrial technologies (WinCCOA SCADA [1]), enhanced with in-house developed frameworks [2,3] allowed for rapid development and effective maintenance of hundreds of complex control system across the organisation. However, this wouldn't be possible without centrally supported IT services and infrastructure. In particular, provisioning of secure network connectivity, computing server hardware, installation of operating system and software, high capacity database service, shared file-systems, consoles in the control rooms or terminal servers for secure remote access have been essential for reliable day-to-day operation. Harmonized efforts of numerous expert teams, ensured the maintenance and upgrades of applications and infrastructure [4] throughout the life-time of control systems.

In general, production control systems are hosted on centrally maintained infrastructure, remain highly homogeneous in terms of technologies and assume the availability of services and also experts. Nevertheless, this may not be achievable for international research collaborations, where the development and precommissioning of the control system may be performed in collaborations, where the development collaborations, where the development and precommissioning of the control system may be performed in a different place than the final location of the equipment. The local infrastructure, available expertise and policies at the deployment site may differ, making the integration and maintenance of such control system seemingly impossible.

LUCASZ CO2 Cooling Stations

LUCASZ [5], *Light Use Cooling Appliance for Surface Zones*, is a medium size I-2PACL CO_2 cooling system developed at CERN with 1kW cooling power at −35°C. It was designed with the purpose of cooling down small and medium sized light tracking detector structures and supporting detector-assembly. A number of research institutes collaborating with CERN needs to build their own replicas of the LUCASZ system to proceed with their activities and commitments related to the phase-II upgrade of the LHC Experiments.

The control system for LUCASZ employs the standard CERN UNICOS [3] technology stack with a Schneider M580 PLC. To facilitate the local operation for non-expert users LUCASZ has a detachable *LocalBOX* featuring an industrial touch-panel. However, its functionalities are limited: for instance it is not suitable for long-term storage of historical data, hence it may not be considered a replacement for a complete, PC-based UNICOS SCADA.

Provisioning a stand-alone cooling system to be operated outside CERN is a technical challenge and requires expertise in several domains not always available in the institutes. Indeed, in addition to cooling, mechanical engineering also the expertise in deployment and configuration of control systems would be required.

PROBLEM AND PROPOSED SOLUTION

Difficulties in deployment and maintenance of control systems, requiring specialized skills, may hinder the effective collaboration and prevent possible re-use of existing instruments or technologies. A solution allowing for the control system to be deployed in form of a "black box" next to the equipment, with the purpose of acting as the *local control station* is demanded.

Unlike for the PLCs, deploying the full UNICOS SCADA application on top of undefined IT infrastructure remains a complex task requiring specific knowledge. In what follows we propose a solution allowing to package a complex SCADA application together with the necessary environment, and at the same time address the problem of the infrastructure compatibility.

Virtual Software Appliances

To address the class of practical problems of lack of local domain-specific technical expertise the IT industry applies a pattern of *software appliances* [6]: self-contained

† Piotr.Golonka@cern.ch

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

assemblies of software tailored and configured for specific use and easy to deploy through well known and practised IT procedures.

Virtual machines (*VM*s) are a widely used method to implement software appliances, supported by IT industry. By virtues of *isolation* and *emulation* they allow to decouple the physical computing infrastructure from the one required by the software embedded in the appliance.

Instances of VMs are created from *VM image files* comprising all the software, the operating system and settings. The image files are the standardized [7] way of distributing the appliances from vendors to end-users.

Local Desktop Virtualisation

Addressing the primary requirement of *local* operation of SCADA, the proposed solution adopts the approach of *local desktop virtualisation* to provide a complete desktop session to the operators using the VM running in a local desktop PC located close to the controlled equipment (see Fig. 1).

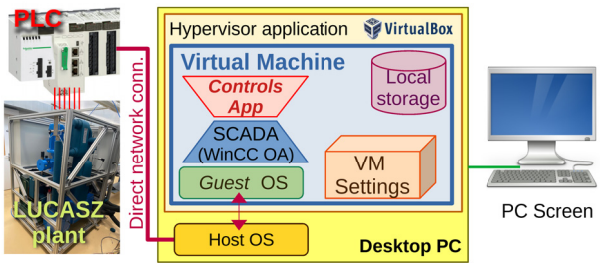


Figure 1: The architecture of proposed solution that employs local desktop virtualisation.

This approach draws from a decade of authors' experience with virtualisation of Linux-based desktop developer environments using the multi-platform and open-source *VirtualBox* hypervisor [8]. Image files containing pre-configured tools and services allow to quickly instantiate new and ready-to-use VMs to be used by developers, in particular those with no experience with configuring a Linux desktop machine. It enables them to develop and test the Human-Machine Interface (HMI) with the user experience similar to real operator consoles, considering aspects such as colour themes, available fonts or pixel-exact rendering. The VMs behave identically regardless of the operating system (Windows, MacOS, various Linux distributions) installed on their desktop machine or a laptop (hence prove the capability of this technology to address the requirement of decoupling the appliance from the hosting infrastructure).

In practice, the perceived performance of VMs hosted in modern desktop machines equipped with SSD storage and enough RAM (extra 4 GB per VM) is most often indistinguishable from the non-virtualised environments. However it is essential to activate the acceleration of virtualisation (VT-d, IOMMU) in the computer's setup – options which are usually disabled by default.

The VM images are based on the currently recommended version of the operating system and applications (CentOS 7, WinCC OA 3.16). New releases containing updates/patches applied are delivered on a regular basis and their import takes around 2 minutes. Desktop session (KDE Plasma) is started immediately after the machine boots, ready to be used in less than 30 seconds. The users appreciate simplicity of use and additional features: auto-adjusting the VM's screen resolution upon its window resize, clipboard synchronization, access to host files, and adequate functionality even without a network connection.

Using these already existing images as a starting point we were able to prototype a "black-box" SCADA appliance for LUCASZ with not much effort. The initial experience clearly indicated that the approach would fit the needs remarkably well, while reusing the already existing work.

Approach to Computer Security

Through the *isolation* features, virtualisation may significantly enhance the security of deployed appliances, if configured correctly. In what follows we assume that the security of the host machine: not only the OS but also the hypervisor software (e.g. VirtualBox) is assured adequately (security and functionality erratas, etc.).

To reduce the potential surface for vulnerabilities the amount of installed software and enabled services in the appliance is reduced to strict minimum. This also allows to reduce the VM startup time and the size of the distributed image (2.5 GB).

The appliance does not require any external *online* resources and could be operated in complete isolation.

A direct connection to the PLC using a dedicated Ethernet network port of the host PC is required. The VM configures it in *bridged mode* for its own use and it should not be available for any other purpose by the host. The MAC address of the network port is pre-configured by the VM, and static non-routable IP addresses are configured for the PLC and the VM's network stack.

If necessary, inter/intra-net connectivity from VM may be enabled: a separate network connection is preconfigured in the VM to provide routed networking through the host's networking stack working in the NAT (*Network Address Translation*) mode. When enabled, it provides the security similar to those of a home router, disallowing incoming connection to the VM, and having all outgoing connection handled by the host's internet connection, respecting all security policies configured by its administrators.

The operation of the appliance is primarily through the main window of the VirtualBox application, which displays the contents of virtualised screen, and passes the mouse and keyboard events to the VM. Remote assistance or operation to the session in which the VM executed may be provided either by the standard tooling or via a commercial *VirtualBox Extension* providing direct RDP access to its console.

The appliance is ready for operation within 30 seconds from its start. Once the VM loads the operating system it auto-starts the SCADA and opens the main HMI Window allowing to visualize the state of the control system. The SCADA-level access control mechanisms are employed requiring the operator to log in to enable commanding on the controlled hardware.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV049

Storage for Historical Values Data

To store the history of the value-changes the SCADA in the self-contained appliance may not assume the availability of large centrally-managed databases, as these typically used at CERN, and needs to fall back to the local file-based mechanism available in WinCC OA. This requires careful parametrisation of data-retention and data-reduction policies to balance the period of available data, its granularity and consumed storage space.

In the proposed solution a dedicated disk image file, with (self-expanding) capacity of up to 100GB is attached to the VM and auto-configures itself to become the storage space for the archive data. The image may be freely detached from one VM and re-attached to another allowing for upgrades of the VM without the loss of archives. In addition, detaching it prior to a VM snapshot allows to exclude the historical data and reduce its size.

Reliability and Recovery Procedure

The reliability of a VM depends of the hardware it runs on. For stand-alone control systems it is not optimal to deploy high-end server hardware. Often a cheaper PC may be sufficient, even though their reliability is lower, provided that a quick and straightforward recovery procedure is available. In case one needs to change the PC, it is sufficient to load the same VM *image* used for the original installation, which takes a few tens of seconds. It is also possible to re-connect the storage of historical values. In addition, standard functionality of VM *snapshots*, which themselves could be exported as images, provide a robust solution for backups.

Update Scenarios

In spite of being a "black box" solution, the appliance needs to be managed through the lifetime of the LUCASZ system. To assure proper support, the major versions of the SCADA and OS of the appliance need to be kept in line with those used at CERN. The envisaged scenario is to prepare and deliver new versions of the image on, with updated components on a regular basis (once per year). The new image may be imported as a new VM working *alongside* the one used in production allowing for its local commissioning and smooth roll-back scenarios. At the moment of writing we have no experience yet with executing an upgrade of the image in production.

Discussion of Alternatives

The presented choice of approach and technologies may not be the only one allowing to address the use case. In what follows, let us present the reasons for not having followed some of other possible solutions.

Remote access to appliances hosted in the on-premises or *cloud* data centres is the most widely known provisioning method for desktop virtualisation. At CERN it is used to provide remote operation of the applications using Terminal Servers hosted on the local "*openstack*" cloud infrastructure.

However, the essential part for the cloud models (Infrastructure/Platform/Software/Desktop as a Service) is the availability of network communication between the user and the hosting site. Concerns about the privacy and non-locality of data may arise due to internal policies. The requirement of vicinity (direct connection) of the self-contained operator station to the operated plant may be hard to address. In addition, the efforts and skills related to provisioning and configuration of such services for a single appliance may exceed those needed by the presented solution.

IIoT and *edge* solutions are very popular trends in industrial controls these days. However, they are not applicableto the discussed problem, as there is no need for storing and processing data in a central place anyway.

Containers are amongst the most popular methods of deployment for reusable software appliances, also applied for Linux-based control systems at CERN [9]. The upcoming enhancements in MS Windows container subsystem allowing to run Linux containers with GUI may become a viable alternative, with advantage of significant reduction in the size of distributed image and startup time, yet at the cost of reduced level of isolation.

Centralized deployment and maintenance of controls applications are being addressed by the SCADA vendors too. For WinCC OA the "*On Premise Administration*" feature (part of *IOT Suite* [10]) has recently become available, yet at the moment of this writing we have no practical experience with it yet.

The fall-back solution is the provisioning of the physical PC, configured with all the software to be installed in the remote location. However, remote expert interventions may be restricted by local IT policies.

Numerous available software products allow for effective virtualisation of GUI desktops in local machines, with efforts for standardization and interoperability [7]. Our choice of Virtual Box was motivated by its availability on numerous platforms and being an open source project. We are convinced that in a general case, other products could fit the same purpose.

EXPERIENCE OF FIRST DEPLOYMENT

The deployment of a virtualised SCADA has been successfully completed for the first time with the LUCASZ installation at DESY in Hamburg, Germany (Fig. 2).

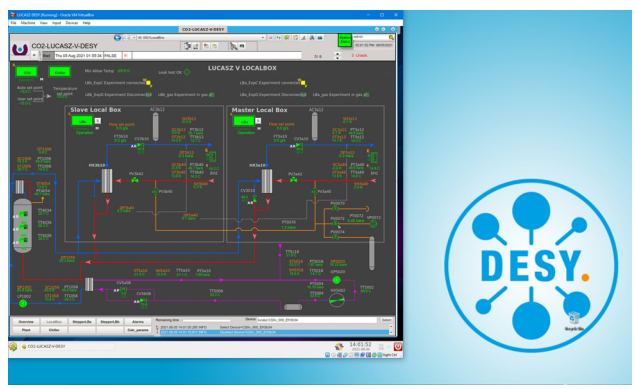


Figure 2: Operator screen of the *CO2-LUCASZ-V-DESY* appliance visible in the VM window of a desktop PC.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV049

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Prior to the deployment, the image of the appliance has been prepared by the experts at CERN. Firstly, a *multipurpose image* with latest versions of SCADA software, OS patches, and a complete generic unparameterized UNICOS application and preconfigured networking parameters (for the PLC communication) was prepared. Based on that the LUCASZ control engineers instantiated a VM, configured the SCADA application (import device instances, I/O addresses, application-specific customizations and operator panels), validated it with a PLC and hardware. Then the VM was exported as the *final image* and delivered to the IT team in the remote institute to be instantiated in the local PC. The same image may be used (with minimal adjustments) for all other instances of the LUCASZ cooling plant.

The initial deployment at DESY resulted in successful operation: after loading and connecting the PLC (with CERN remote assistance), the communication with SCADA was established immediately. The deployment and hand-over to the local team for operation was completed in less than a day, which is very encouraging for the future.

APPLICABILITY

The presented solution addresses primarily the need for an easy-to-deploy standalone local control system to be used for replicas of highly specialized equipment, built from engineering *blueprints.* However, the scope of the use for this approach is wider.

Large and complex research equipment is often built through a collaborative effort partitioned to work-packages distributed worldwide. Some of the contributions may be delivered as complete "turn key" systems to bring also the *operational experience* (e.g. gas system for KEK contributed by CERN). Their controls technologies may differ from the ones used widely by the collaboration whereas re-engineering is not feasible. Virtualised appliances allow to overcome this inhomogeneity and ease the integration.

Recently the appliance has been used to implement a small portable laboratory setup at CERN for tests of CO_2 long vertical transfer lines recording data from a few pressure sensors wired to a PLC. It took less than 3 hours to set up the PLC and the SCADA appliance running in a laptop. Numerous CERN groups have already expressed their interest for this type of portable SCADA deployments to be used in instrumentation of test stands.

CONCLUSIONS AND OUTLOOK

The use of virtualisation to provision self contained SCADA systems in form of easy-to-deploy software appli-ances has been carried out successfully for the first remote instance of LUCASZ system in DESY, with subsequent ones already scheduled for NIKHEF and INFN. The same approach is envisaged to provide control system for the Gas System to be contributed by CERN to the T2K-ND280 experiment in Japan. After a decade of the internal use of desktop virtualisation for software development, we found the approach to be effective for production and lab setups to deliver the full functionality of SCADA. We expect it to be used more and more not only internally at CERN but also for contributions to international research collaborations.

We believe that the presented solution is generally applicable also for other control system technologies and will further enable the reuse of unique hardware across research community, also beyond physics.

REFERENCES

[1] Siemens Simatic WinCC Open Architecture, www.winccoa.com

[2] O. Holme, M. Gonzalez Berges, P. Golonka, S. Schmeling, "The JCOP framework", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper WE2.1-6O.

[3] E. Blanco, F. B. Bernard, P. Gayet, H. Milcent, "UNICOS: An Open Framework", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper THD003, pp. 910-912.

[4] R. Kulaga, J. Arroyo Garcia, M. Boccioli, E. Genuardi, and P. Golonka, "Large-scale upgrade campaigns of SCADA systems at CERN – organisation, tools and lessons learned" in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct 2017, pp. 1384-1388. doi:10.18429/JACoW-ICALEPCS2017-THPHA021

[5] L. Zwalinski *et al.*, "First steps in automated software development approach for LHC Phase II upgrades CO_2 detector cooling systems", in *Proc. ICALEPCS'19* , New York, USA, Oct 2019, pp. 1488-1491. doi:10.18429/JACoW-ICALEPCS2019-WEPHA170

[6] "Virtual Appliances: A New Paradigm for Software Delivery", VMWare whitepaper, https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vam/vmware-virtual-appliance-solutions-white-paper.pdf

[7] Open Virtualization Format, DTMF specification, ISO-17203, https://www.dmtf.org/standards/ovf

[8] VirtualBox cross-platform virtualisation application, www.virtualbox.org

[9] R. Voirin, T. Oulevey, and M. Vanden Eynden, "The state of containerization in CERN Accelerator Controls", presented at ICALEPCS'21, Shanghai, China, October 2021, paper THBL03, this conference.

[10] Siemens Simatic WinCC OA IOT Suite, https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada/simatic-wincc-oa/simatic-wincc-oa-iot-suite.html

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL01

THE Laser MegaJoule FACILITY STATUS REPORT

H.Cortey, CEA/DAM CESTA, Le Barp, France

Abstract

The Laser MegaJoule (LMJ), the French 176-beam laser facility, is located at the CEA CESTA Laboratory near Bordeaux (France). It is designed to deliver about 1.4 MJ of energy on targets, for high energy density physics experiments, including fusion experiments. The first bundle of 8-beams was commissioned in October 2014 [1]. By the end of 2021, ten bundles of 8-beams are expected to be fully operational.

In this paper, we will present:

- The LMJ Bundles Status
- The main evolutions of the LMJ facility since ICALEPS2019
- The result of a major milestone for the project: 'Fusion Milestone'

INTRODUCTION

The laser Megajoule (LMJ) facility, developed by the "Commissariat à l'Energie Atomique et aux Energies Alternatives" (CEA), is designed to provide the experimental capabilities to study High Energy Density Physics (HEDP). The LMJ is a keystone of the Simulation Program, which combines improvement of physics models, high performance numerical simulation, and experimental validation, in order to guarantee the safety and the reliability of French deterrent weapons. When completed, the LMJ will deliver a total energy of 1.4 MJ of 0.35 µm (3ω) light and a maximum power of 400 TW.

The LMJ is sized to accommodate 176 beams grouped into 22 bundles of 8 beams. These will be located in the four laser bays arranged on both sides of the central target bay of 60 meter diameter and 40 meter height. The target chamber and the associated equipment are located in the center of the target bay.

The first bundle of eight beams has been commissioned at the end of 2014. The second bundle has been commissioned at the end of 2016 following the same commissioning process. Seven additional bundles are now operational since the end of 2020, and the first physics experiments using the 56 operational beams took place in the second semester of 2019.

The PETAL project consists in the addition of one short-pulse (0.5 to 10 ps) ultra-high-power (1 up to 7 PW), high-energy beam (1 up to 3.5 kJ) to the LMJ facility. PETAL offers a combination of a very high intensity petawatt beam, synchronized with the nanosecond beams of the LMJ [2].

The first phase of nuclear commissioning of LMJ has been achieved to take into account high-energy particles created by PETAL, and neutron production from D-D fusion reaction. A subsequent phase will take into account tritium targets.

This paper describes the LMJ facility status and the new target diagnostics. A milestone physics experiments is presented to illustrate the facility capacity to realize the first thermonuclear fusion by implosion of a D_2 capsule in a cavity.

LMJ BASELINE

The 176 beams ($37 \times 35.6\,cm^2$ each) are grouped into 22 bundles of 8 beams. In the switch-yards, each individual bundle is divided into two quads of 4 beams, the basic independent unit for experiments, which are directed to the upper and lower hemispheres of the target chamber.

Basically, an LMJ laser beam line is composed of three parts: the front-end, the amplifying section, the switch-yard and the final optics assembly.

The front end delivers the initial laser pulse (up to 500 mJ). It provides the desired temporal pulse shape and spatial energy profile.

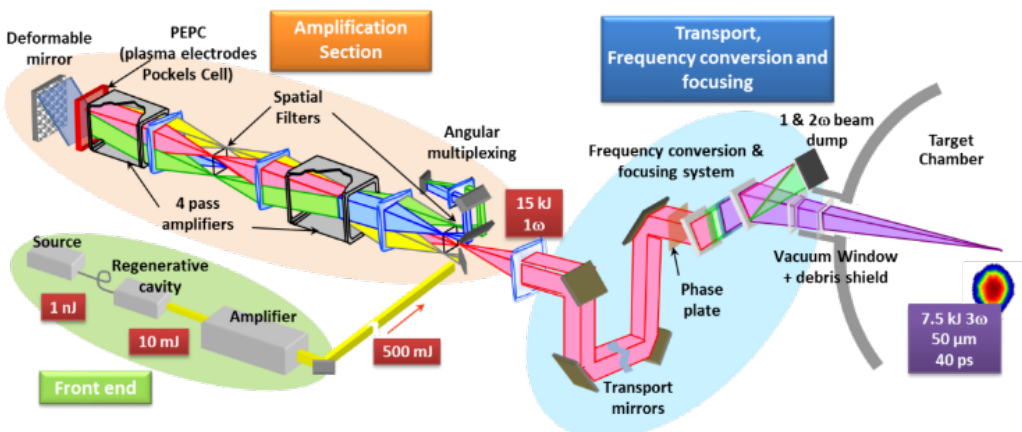


Figure 1: Laser beamline schematic diagram.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

The initial pulse leaving the front end is amplified four times through two amplifiers, in order to obtain the energy required for the experiments (up to 15 kJ at 1 ω per beam). Positioned between the two amplifiers, focusing lenses, associated to a diaphragm (spatial filter pinhole), take out the parasitic (noise) beams that may arise. Beyond the two amplifiers is a reflecting mirror (M1), making the four passes possible through angular multiplexing, as shown on Fig. 1. The surface of this mirror is deformable (being controlled by electro-mechanical actuators), allowing beam wavefront distortions to be controlled.

The 8 beams coming from the amplification section are divided into two quads. Each quad is transported over more than 40 meters into the target bay and is directed to the upper or lower hemisphere of the target chamber using six transport mirrors per beam. Each quad arrives into a frequency conversion and focusing system (SCF). Inside the SCF, the beams frequency is changed from infrared (1,053 nm) to ultraviolet (351 nm). The frequency conversion is realized by two KDP crystals, with a global efficiency of about 60%. The beams are focused on target using a 3 ω focusing grating. This specific component allows spectral separation (to ensure only 351 nm wavelength reaches the target).

LMJ BUNDLES STATUS

The completion of the LMJ facility (176 operational beams, full target bay equipment, …) requires a long period of time. During this period, we need to assemble the bundles, commission the assembled bundles, but also realize experiments addressing different physics domains with the operational bundles, in order to control the ignition process.

Today, Nine LMJ bundles (18 quads) are operational since the end of 2020. Before the end of the year, one more bundle will be operational and four more bundles should be assembled. At the end of 2021, 80 beams will be operational.

LMJ & PETAL TARGET DIAGNOSTICS

Fourteen target diagnostics are now operational around the target chamber, and two new diagnostics will be operational at the end of 2021. Table 1 summarizes the different functions and the main features of these diagnostics. Some diagnostics are insertable using a SID (System for Diagnostics Insertion). A SID is a telescoping system that provides a precise positioning of a diagnostic close to the center of the target chamber. It moves a 150 kg diagnostic with 50 µm precision. Four SID are operational: three in the equatorial plane of the target chamber, and one in a polar position on the upper hemisphere. Three more equatorial SID will be operational at the end of 2021, in 2022 and 2023.

Table 1: Target Diagnostics Description

Diagnostic	Function(s)	Insertable	Operational
GXi-1	Hard gated X-ray middle resolution imager	Yes	Yes
DMX	Multi-channel X-ray diagnostic: broadband X-Ray spectrometer + laser entry hole imager & soft-X-ray spectrometer	No	Yes
mini-DMX	Broadband X-Ray spectrometer	Yes	Yes
SSXi	Soft X-ray spectro-imager	Yes	Yes
GXi-2	Large field of view Hard X-Ray imager	Yes	Yes
SPECTIX	Hard X-Ray spectrometer	Yes	Yes
CRACC	Proton radiography (part of SEPAGE)	Yes	Yes
SESAME 1&2	Electron spectrometers	No	Yes
SEPAGE	Electron & ion/proton spectrometer & proton radiography	Yes	Yes
FABS	Full aperture scattering measurement station	No	Yes
SHXI	Streaked Hard X-Ray Imager (time resolved 1D image)	Yes	Yes
Neutron pack	Neutron yield measurement, neutron Time of Flight measurement	No	Yes
EOS pack	Velocity measurement (Visar), Shock Break Out/pyrometers	Yes, partly	Yes
DEDIX	Heterodyne velocimetry	Yes	Yes
Nbi	Near Backscattering Imaging System	No	2021
UPXi	LEH imager	No	2022
HRXi	High Resolution X-Ray imager	Yes	2022
HRXS	High resolution X-Ray spectrometer	Yes	2023
HRXi	High Resolution X-Ray imager	Yes	2022

PETAL: A PETAWATT CLASS LASER INSIDE THE LMJ FACILITY

The PETAL project consists in the addition of one short-pulse (0.5 to 10 ps) ultra-high-power (1 to 7 PW), high-energy beam (1 to 3.5 kJ) to the LMJ facility (see Fig. 2). The PETAL beam is focused in the equatorial plane of the target chamber.

PETAL Architecture

PETAL laser is based on the chirped pulse amplification (CPA) technique combined with optical parametric amplification (OPA). The front end consists in a standard Ti:Sap-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL01

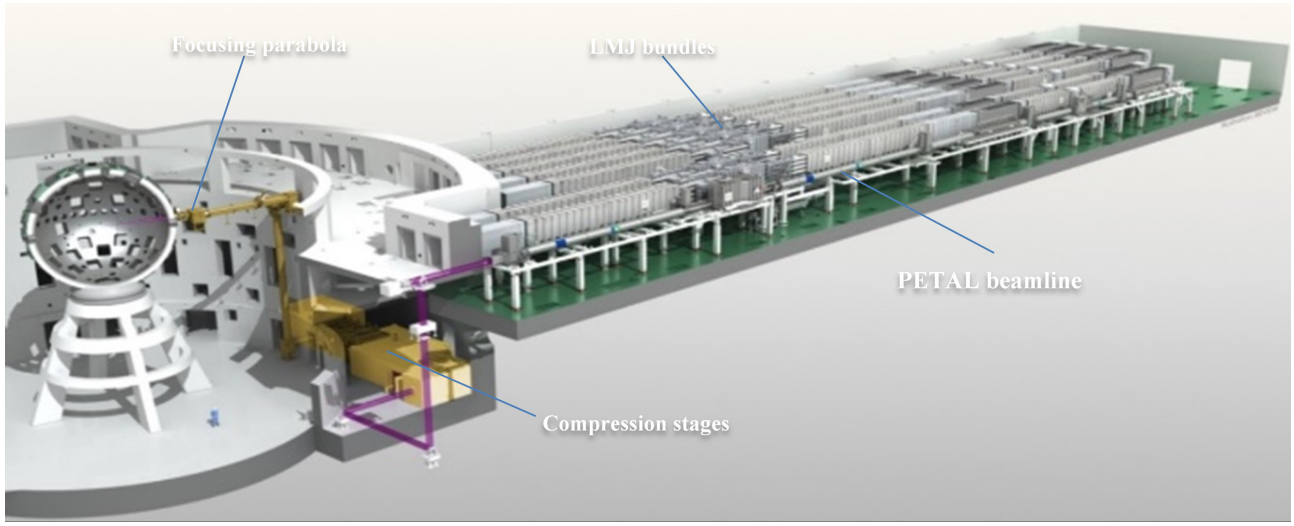


Figure 2: Implementation of PETAL. The PETAL beam is focused in the equatorial plane of the target chamber.

phire mode locked oscillator delivering 3 nJ, 100 fs, 16 nm pulse at 77.76 MHz and 1053 nm wavelength. The initial pulse (3 nJ, 100 fs,16 nm) is stretched to 9 ns in an Offner stretcher in eight passes. Then the pulse is sent to the Pre-Amplifier Module (PAM) including OPA stages and pump laser to reach 150 mJ.

The PETAL amplifying section has the same architecture as the LMJ using a single beam instead of the 8 beams of an LMJ line. It uses 16 amplifier laser slabs arranged in two sets and delivering up to 6 kJ (1.7 ns, 3 nm). The main differences with the LMJ amplifying section baseline are the wavefront and chromatism corrections.

The compression scheme is a two-stages system. The first compressor, in air atmosphere, reduces the pulse duration from 1.7 ns to 350 ps. The output mirror is segmented in order to divide the initial beam into 4 sub-apertures which are independently compressed and synchronized into the second compressor under vacuum. These sub-apertures are coherently added using the segmented mirror with three interferometric displacements for each sub-aperture. The pulse duration is adjustable from 0.5 to 10 ps.

After a transport under vacuum, the beam is focused in the equatorial plane of the LMJ chamber via an off-axis parabolic mirror with a 90° deviation angle, followed by a pointing mirror.

PETAL Performances

The first high energy test shots in the compressor stage of PETAL were performed in May 2015. They demonstrated the Petawatt capabilities of PETAL with a 1.15 PW power shot (850 J energy and 700 fs duration) [3]. This Petawatt power has then been brought to the LMJ target chamber center in December 2015, and a test shot coupling LMJ and PETAL has been performed at the same date.

The commissioning of focal spot on target and the main performance on target has been performed during the first campaigns (2nd semester 2017 to 1st semester 2019), see Fig. 3.

It concerns:

- The alignment of the sub-aperture compression stages in order to optimize the pulse compression (the mean duration was 685 fs for the 2018 shots in ps regime, the best 2018 value being 570 fs),
- the intensity on target of 9.2×10^{18} W/cm² in 2019 (intensity inferred from measurements at the end of the compressor for the 358 J @ 690 fs shot and confirmed by the TwIST measurement [4],

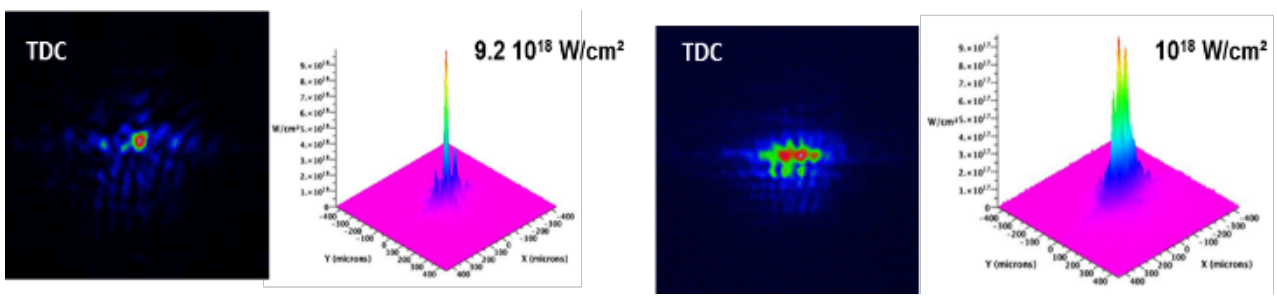


Figure 3: Focal spot (left) for the 358 J @ 690 fs shot giving 9.2×10^{18} W/cm², focal spot (right) optimized for the shot on the 25 µm wire with 396 J @ 3 ps and 10^{18} W/cm². They are measured on the compression diagnostic table TDC during shot.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- the alignment performance (positioning, pointing) and demonstration of the first associated LMJ and PETAL laser shots on target.

Specific optimizations have been made to deliver 10 ps pulses on target and to shot for the 1st time on PETAL on a 25 µm wire.

The next improvement of PETAL performances will concern:

- the upgrade of optics under vacuum in order to increase the transported energy (ongoing action),
- the improvement on damage threshold,
- the temporal contrast measurement
- and the focal spot optimization and characterization.

EXPERIMENTS: "FUSION MILESTONE"

At the end of 2019 a major milestone has been reached with the fusion milestone. It was the first fusion experiments in cavity with 6 laser bundles in an indirect drive scheme. By this way, we are able to provide more than 150 kJ and 55 TW on target. Goals of this experiment were

- the realization of a thermonuclear fusion by implosion of a D_2 capsule in a Rugby cavity,
- the measurement of produced neutrons,
- the design robust to laser illumination asymmetry,
- the good capacity of prediction from our simulations.

The experiment is described on Fig. 4. The ten experiments provided a rich set of measurements for the validation of the 3D simulation code. Some of the experimental results are presented on Fig. 5.

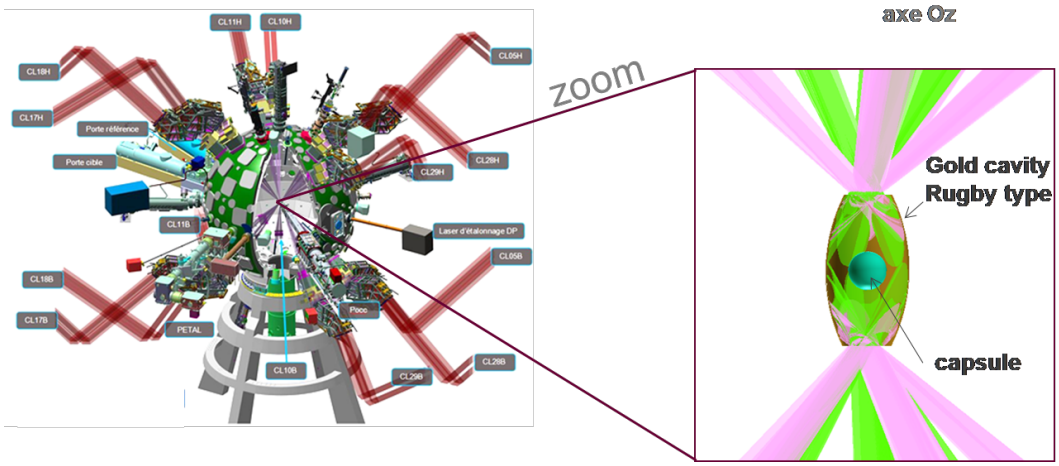


Figure 4: Experimental set-up.

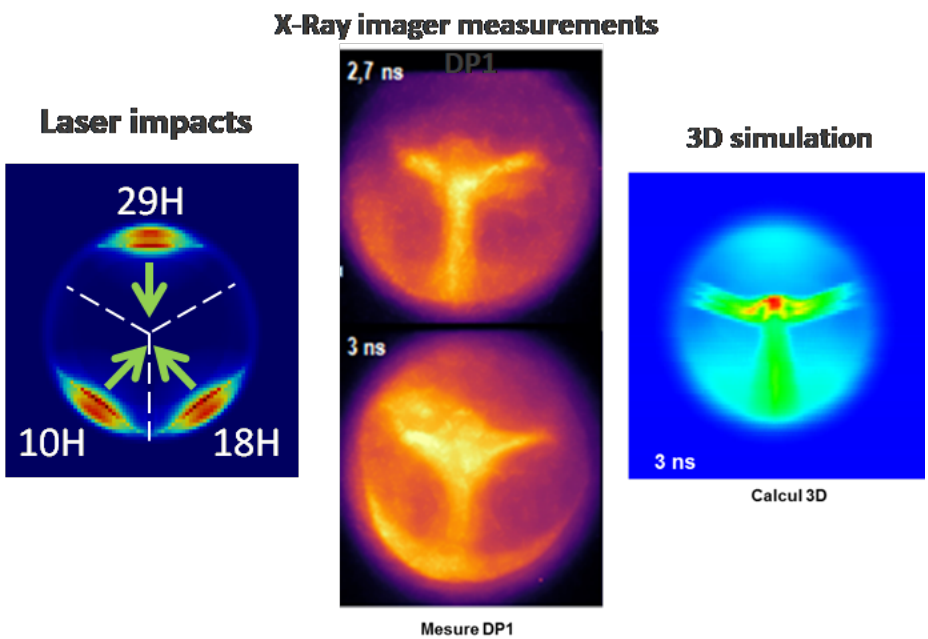


Figure 5: Hydrodynamics measurement of the cavity walls.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL01

CONCLUSION

At the end of 2021, The LMJ facility will be operational with ten LMJ bundles (80 beams) and with PETAL, a Petawatt class laser.

The combination of PETAL and LMJ extends the versatility of the laser facility.

From its commission in 2014, the capacity of the facility is constantly growing up. The assembly of new bundles go on following a rhythm of 2 bundles per year.

REFERENCES

[1] P. Vivini and M. Nicolaizeau, "The LMJ: Overview of recent advancements and very first experiments," *Proc. SPIE*, vol. 9345, pp. 1–8, 2015. doi:10.1117/12.2079812

[2] J.-L. Miquel, "LMJ & PETAL Status and Program Overview," *Journal of Physics: Conference Series*, vol. 717, p. 012084, 2016. doi:10.1088/1742-6596/717/1/012084

[3] N. Blanchot *et al.*, "1.15 PW–850 J compressed beam demonstration using the PETAL facility," *Opt. Express*, vol. 25, pp. 16957-16970, 2017. doi:10.1364/OE.25.016957

[4] R. Wrobel, "Development of LMJ and PETAL plasma diagnostics: present status," LMJ/PETAL User Meeting, 2018.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

DISCOS UPDATES

S. Poppi*, M. Buttu, G.Carboni, A.Fara, C. Migoni INAF - OA Cagliari, [09047] Selargius, Italy
M. De Biaggi, A. Orlati, S. Righini, F. R. Vitello, INAF - IRA, [40138] Bologna, Italy
M.Landoni[1], INAF - Osseravatorio Astronomico di Brera, Merate, Italy
[1]also at INAF - OA Cagliari

Abstract

DISCOS is the control software of the Italian Radio Telescopes and it is based on the Alma Control Software. The project core started during the construction of the Sardinia Radio Telescope (SRT) and it further developed to support also the other antennas managed by INAF (National Institute for Astrophysics), which are the Noto and the Medicina antennas. Not only DISCOS controls all the telescope subsystems - like servo systems, backends, receivers and active optic system - but it also allows users to exploit a variety of observing strategies. In addition, many tools and high-level applications for observers have been produced over time. The development of this software follows test-driven methodologies, which, together with real hardware simulation and automated deployment, speed up testing and maintenance. We here describe the status of the DISCOS project and of the related activities, also presenting its ongoing upgrades.

INTRODUCTION

The Italian National Institute for Astrophysics (INAF) manages three single-dish radio telescopes: The Sardinia Radio Telescope (SRT), the Noto and the Medicina radio telescopes. These are open sky facilities; the international scientific community is invited to submit observing projects through calls for proposal, published twice a year [1]. The telescopes cover radio bands from 305 MHz up to 26.5 GHz, allowing many research topics to be explored. Examples are pulsars, astrochemistry, extragalactic sources, space weather. SRT and Noto are already provided with an active surface, allowing for observations at much higher frequencies; Medicina is planned to have it installed within spring 2023.

The control software plays a key role in an observing facility, allowing the users to perform the needed observations by using proper strategies and modes, while ensuring the quality of the acquired data. Therefore, in 2004 we started developing NURAGHE, the SRT control software. In 2007 we parallely began the ESCS (Enhanced Single-dish Control Software) project, devoted to the Medicina and Noto radio telescopes. Eventually, in order to optimize the efforts, in 2015 the three development lines were unified in DISCOS, a common control software for all the three telescopes.

DISCOS is built on top of the Alma Common Software, which is based on CORBA [2]. This framework allowed us to realize a modular software mostly made of common codebase, reused and deployed at all sites, as much as possible. Considering this, only a small part of the codebase (23%) is telescope-specific, essentially in the low-level and no-logic control of the devices and of the telescope hardware [3]. In 2017 we refactored part of the code, in order to adapt it to the upgrade of the framework to a newer version of ACS [4]. Also, we chose Github [5] to track issues and manage version control.

Equally important is the development strategy. We followed guidelines in the adoption of an approach called Behavior Driven Development (BDD) [6] which aims to test the software behaviour and it is used together with Test Driven Development (TDD) and unit testing strategies [7].

In the following sections we show the ongoing and planned DISCOS upgrades. In particular we present the integration of new instrumentation, such as new receivers and backends, a new simulated environment of the SRT hardware devices, a middleware DISCOS wrapper called SURICATE and a simple web-based monitoring and alarm system.

TELESCOPES UPGRADE

In 2019 INAF was granted a PON (National Operational Program) funding to upgrade the Sardinia Radio Telescope and its infrastructure toward higher frequencies (up to 100 GHz). Within this scope, up to a 15% of the overall funding was aimed to also upgrade the Medicina and Noto radio telescopes. The funded project includes the acquisition and the installation of new receivers on the telescopes:

- Three coaxials receivers K/Q/W band (18–26 GHz, 34–50 GHz, 80–116 GHz). for Medicina, Noto and SRT.
- 16-Beam W Band receiver (70–116 GHz) for SRT
- 19-Beam Q Band receiver (33–50 GHz) for SRT
- A bolometric millimetre camera for SRT operating in the 77–103 GHz made of 408 detectors.

Furthermore, the PON project includes, as concerns the SRT, the procurement of three digital data acquisition systems (backends), a new state-of-the-art metrology system, an HPC system and laboratory instrumentation. Also, the telescope minor servo system, which is responsible for the proper positioning of the telescope optics, will undergo a refactoring and a major upgrade.

It is worth noting that new receivers and new backends will need a big effort in order to integrate them in the control software. To do this, we exploited the ACS architecture and the DISCOS modularity. Not only the new receivers have the same interfaces, but also the communication protocols are the same of the SRT first-light receivers.

* sergio.poppi@inaf.it

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

The availability of new instrumentation will allow users to exploit additional observing modes, so we are also upgrading BASIE (the schedule creator), which is a fundamental tool to plan and execute the observations. BASIE was designed to create the schedule files required by DISCOS to carry out continuum and spectroscopy observations, according to different strategies - like on-the-fly scanning, on-off and nodding [8]. Thanks to this tool, users do not need to deal with the writing of the complex schedule files: they are only asked to specify a combination of receiver and backend, providing basic information on the celestial sources to be observed, the desired strategy and the configuration of the devices.

SIMULATORS

Writing a simulator helps the developers in writing more reliable code for the actual control software of the radio telescope, the DISCOS control software. Being able to test the code without having to rely on the hardware represents a huge advantage in the development and maintenance processes. It speeds up the development of new features and the bug fixing, keeping under control the codebase through the adoption of continuous integration practices. Both unit and functional tests are cleaner and shorter, no code mocking is required, they work both over simulators and real hardware. In addition, the integration of new components is easier and more reliable, and the hardware itself can be verified by running the simulator tests over it. A suite of simulators, capable of reproducing different scenarios, can be exploited to write and execute a great variety of tests whenever a modification to the control software code gets pushed to the main repository. All these considerations led us to change our process to integrate new devices, so that the writing of its simulator is now the mandatory first step. To improve and speed up our capability to add simulators to our environment, we developed a simulator framework. The framework for the simulators in written in Python and is composed of two layers. The topmost layer is in charge of handling network communications, it behaves as a server, listening for incoming connections from clients and relaying every received byte to the other layer of the framework, the simulation layer. This layer is where received packets are parsed and executed. It can faithfully replicate the behavior of the hardware, or it can simply act as a protocol interpreter by providing back proper answers. [9]

Furthermore, the framework allows to test how the control software code reacts under expected error conditions. In fact, it provides an easy way to simulate unlikely scenarios that are very difficult or, in some cases, impossible to replicate by only using the hardware. This allows the developers to write more reliable and robust code, likely permitting the recovery from an error condition without having to resort to a complete reboot of the system.

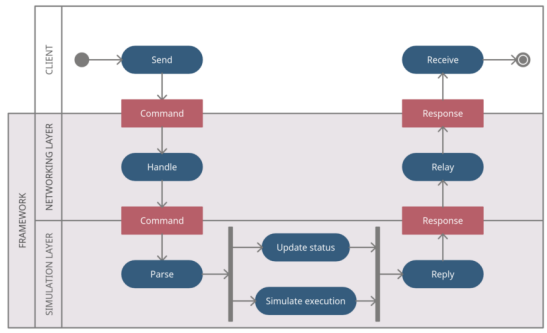


Figure 1: Simulators framework layers and communication behavior.

DEPLOYMENT

An automatic Ansible [10] pipeline creates users, configures the network, sets up the shared file system, installs all the required dependencies (i.e.: Alma Common Software), installs all the required observing tools and finally the DISCOS software, creating a fully-configured working environment. Having an automated deployment means that both development and production environments are aligned, all the process is automatically documented and the system can be easily updated and restored. Not only Ansible allows the deployment on physical machines, but it can also provide, through Vagrant, virtual machines based on Oracle VirtualBox.

A DISCOS installation is basically composed of:

- A manager hosting ACS services and maintenance tools
- A console, providing the user interfaces and accounts for all the allowed observing projects
- Storage cluster, mounted with LustreFS

The deployment scripts allow to select which telescope the installation is specific for and which branch or the tag from the Github repository is to be provided [11].

In summary, all the DISCOS instances are well aligned among all the installations. This allows us to speed up maintenance and bug tracking.

SURICATE

Suricate [12] is a middleware which exposes APIs to the clients and offers an abstraction from the control system and the programming language. Moreover, it allows DISCOS to be easily extended bypassing the framework on which it is based, getting advantage of new technologies.

It is composed of a sampler that collects from DISCOS the telescope status, its configuration and relevant parameters, writing them in on a in-memory database (redis-db). Then a different module, the db-filler, gets the data from the redis-db to save them into a persistent SQL database. A HTTP server gets requests from clients and reads the values from the database, returning a JSON answer. Also, the HTTP server

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

receives user commands from clients, forwarding them to DISCOS.

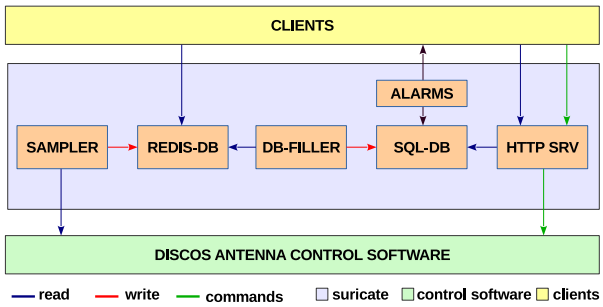


Figure 2: Suricate is a middleware which offers API and abstraction from the operating system and the programming language.

CONCLUSIONS

DISCOS is continuously evolving as new instrumentation is installed on the Italian radio telescopes. One of the major upgrades we are performing is the integration of new high-frequency receivers and data acquisition backends into the control software.

Such upgrades are made easier by the development choices the team previously made. The automatic deployment and software provisioning, together with the hardware simulators, allow us to add new features while keeping software robustness, speeding up maintenance and bug fixing. The Suricate middleware, thanks to its abstraction from the underlying control software, allows us to produce new applications (e.g. GUIs) exploiting also technologies that are not supported by the ACS framework.

ACKNOWLEDGEMENTS

The Sardinia Radio Telescope is funded by the Ministry of University and Research (MIUR), Italian Space Agency (ASI), and the Autonomous Region of Sardinia (RAS) and is operated as National Facility by the National Institute for Astrophysics (INAF). The Medicina and Noto radio telescope are funded by the Ministry of University and Research (MIUR) and are operated as National Facility by the National Institute for Astrophysics (INAF).

The Enhancement of the Sardinia Radio Telescope for the study of the Universe at high radio frequencies is financially supported by the Programma Operativo Nazionale (PON) del MIUR "Ricerca e Innovazione 2014-2020" Avviso D.D. n° 424 del 28/02/2018/ per la concessione di finanziamenti finalizzati al potenziamento di infrastrutture di ricerca, in attuazione dell'Azione II.1 - Proposta Progettuale PIR01_00010.

REFERENCES

[1] https://www.radiotelescopes.inaf.it/.

[2] G. Chiozzi et al., "The ALMA common software: a developer-friendly CORBA-based framework", in *Proc. SPIE*, vol. 5496, p. 205, September 2004.

[3] A. Orlati, M. Bartolini, M. Buttu, A. Fara, C. Migoni, S. Poppi, and *et al.*, "Design Strategies in the Development of the Italian Single-dish Control System", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, paper MOPGF110, pp. 330–333, http://jacow.org/icalepcs2015/papers/mopgf110.pdf, doi:10.18429/JACoW-ICALEPCS2015-MOPGF110, 2015.

[4] A. Orlati, M. Bartolini, M. Buttu, A. Fara, C. Migoni, S. Poppi, and *et al.*, "Evolution in the Development of the Italian Single-dish COntrol System (DISCOS)", in *Proc. ICALEPCS2017*, https://doi.org/10.18429/JACoW-ICALEPCS2017-THPHA014, 2018.

[5] DISCOS repository, https://github.com/discos

[6] M. Buttu *et al.*, "Rules of thumb to increase the software quality through testing", in *Software and Cyberinfrastructure for Astronomy III*, vol. 9913, p. 99130B, 2016. doi:10.1117/12.2230626

[7] Bartolini *et al.*, "DISCOS Project Status and Evolution Towards Continuous Integration", in *Astronomical Data Analysis Software and Systems XXVI ASP Conference Series*, Vol. 521, 2019.

[8] Righini S., Bartolini, M., 2016, Basie User Manual, IRA Technical Reports 492, http://www.ira.inaf.it/Library/rapp-int/492-16.pdf

[9] Carboni G., and Buttu M., "Discos simulators documentation", *INAF Technical Reports 82*, 2021. doi:10.20371/INAF/TechRep/82

[10] Ansible web site, https://www.ansible.com/.

[11] https://github.com/discos/deployment

[12] https://github.com/discos/suricate

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

CRYOGENIC CONTROLS TODAY AND TOMORROW

M. Pezzetti, P. Gayet, CERN, Geneva, Switzerland

Abstract

The CERN cryogenic facilities demand a versatile, distributed, homogeneous and highly reliable control system. For this purpose, CERN conceived and developed several frameworks (JCOP, UNICOS, FESA, CMW), based on current industrial technologies and COTS equipment, such as PC, PLC and SCADA systems complying with the requested constraints. The cryogenic control system nowadays uses these frameworks and allows the joint development of supervision and control layers by defining a common structure for specifications and code documentation. Such a system is capable of sharing control variables from all accelerator apparatus. The first implementation of this control architecture started in 2000 for the Large Hadron Collider (LHC). Since then, CERN continued developing the hardware and software components of the cryogenic control system, based on the exploitation of the experience gained. These developments are always aimed to increase the safety while improving, at the same time, the performance. The final part will present the evolution of the cryogenic control toward an integrated control system SOA based, using the Reference Architectural Model Industrie 4.0 (RAMI 4.0).

INTRODUCTION

The CERN cryogenic infrastructures for accelerators, detectors and test systems include large and complex facilities able to cool the equipment down to 80 K with liquid nitrogen (LN_2), to 4.5 K with liquid Helium (*LHe*) and to 1.9 K with super fluid helium. These facilities are placed in dedicated test facilities, experimental areas and around the 27 km LHC. The complexity of the cryogenic facility requires an automated and homogeneous control system that must be flexible, reliable and distributed around the entire CERN LHC cryogenic line QRL [1]. Today, for the High luminosity (HL-LHC), era some adaptions will be applied to fully automate the control development and to prepare the integration of the control system with the existing Maintenance Management Software (MMS) and the operational support environment from the conception stage to the daily operation. Tomorrow, a new generation of large accelerators is forecasted, making the story far from finished. Hence this article will also present additional thoughts and potential developments proposals.

CRYOGENIC CONTROL SYSTEM TODAY

Currently, the cryogenic control system follows the standard automation pyramidal structure of the International Electrotechnical Commission (IEC-62264) and is based on industrial components deployed in all control layers (see Fig. 1):

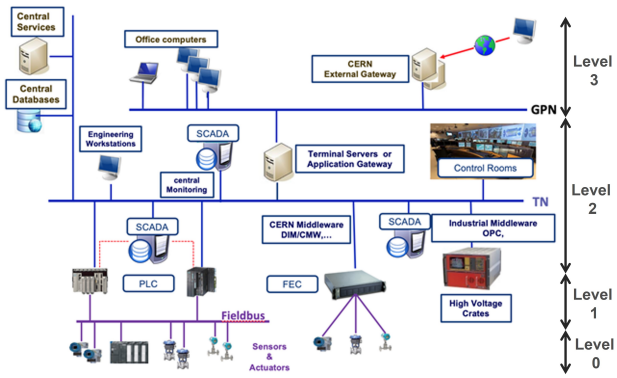


Figure 1: CERN Industrial Control Ecosystem.

- Instrumentation layer (IL): The cryogenic instrumentation needs a significant number of industrial sensors to collect data, electronic conditioning units and actuators to execute commands. To ensure the correct communication with the devices, both copper cable connections and a protected and dedicated Ethernet network or industrial field-buses are used. In the LHC, where the environment is hostile, radiation resistant instruments and fieldbuses were chosen.
- Control layer (CL): The control duties are executed within PLC, with safety interlocks either cabled in the electrical cabinets or programmed in local protection PLCs. In radiation exposed area radiation tolerant crates and field-bus are used, coupled with standard CERN Front End Computer (FEC). The long-distance integration site to site or toward the supervision layer, relies on the CERN Ethernet Technical Network (TN) whereas the local ones (internal to a cryogenic site) are implemented on field-buses using both fibres and copper cables or direct cabled to the cabinets.
- Supervision layer (SL): All cryogenics systems are supervised through Data Servers (DS) running the WinCC-OA® SCADA system within off-the-shelf's Linux machines. The Human-Machine Interface (HMI) clients allows operators to monitor and act on the cryogenics facilities using Linux PC deployed within the control rooms. In addition to the classical SCADA features, several functionalities have been added in the last decade: visualisation of the process hierarchy, access to the interlocks per devices, control loop auto tuning, direct access to device documentation, etc. This layer provides also interfaces to the Management Layer, and a dedicated connections toward the CERN central alarm system and the LHC experiments control systems.
- Maintenance & Operation Management System layer (ML): The central long-term logging database (NXCALS), the CERN CMMS (Infor-EAM) and the Accelerator Fault Track application (AFT) are parts of this

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

layer. They monitor the whole process and integrate it into the campus environment. These applications are currently being integrated with the present cryogenic control system and the next run will be a good opportunity to validate them.

- The Company Overall Management layer (CL): At CERN, this level is filled with Administrative Information Services (AIS) applications. Up to now there are few ad-hoc integrations with ML components for specific CERN groups. However, the complete vision is lacking and there is not yet a well coordinated development.

Cryogenic Control Applications

The cryogenic process control system uses the first implementation of the UNified Industrial Control System (UNICOS), a CERN-made framework to develop of industrial control applications that integrates the development the SL and the CL in PLC and SCADA as it's done in a Distributed Control System (DCS). The main benefit of UCPC applications (UNICOS coupled with the Continuous Process Control package) [2, 3] is to ease the operator's ability to follow the evolution of the process and to understand the dynamics of a situation as well as the role of each physical component. It also helps to identify the origin of a failure, predict what could happen in the near future, etc. To achieve these duties, operation teams have access to classical SCADA tools (process synoptics, time stamped alarms, events lists, trend curves with long term storage, etc.), but also to the visualisation of the process hierarchy, the interlocks per devices, the control loop auto tuning, and the device documentation, etc.

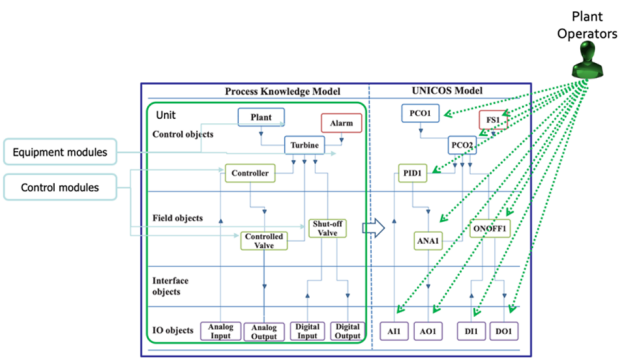


Figure 2: cryogenic device hierarchy according to IEC61512-1 decomposition and its equivalence in UCPC.

The UCPC model is based on the IEC61512-1 model that decomposes (see Fig. 2) the cryogenic plant in smaller components (Equipment Modules (EM) or Process Control Object (PCO) for UCPC) down to the smallest possible equipment module. These EM are structured in Control Modules (CM) representing the sensors and actuators or regulators and alarms connected to the process through inputs and outputs. The UCPC device model permits the concurrent access from the supervision and process control hierarchy to the devices with the mechanism as exposed in Fig. 3. It allows the operators to intervene at the right place without side effects to the other components.

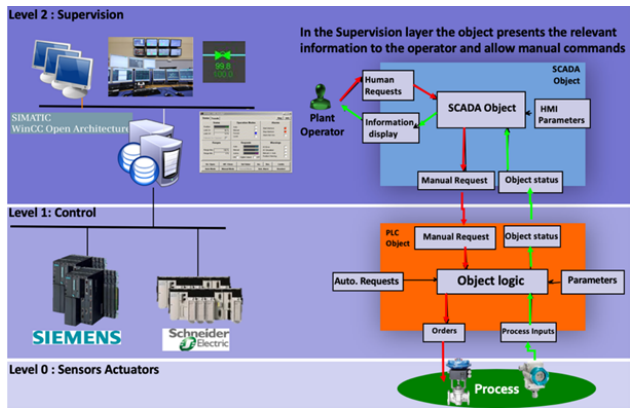


Figure 3: UCPC Control & supervision layer integration.

Application Development

To achieve the delivery of high-quality control system applications within a certain cost and time constraints, a generator, using a step-wise development methodology (as shown in Fig. 4), has been elaborated by CERN. This 5 part process makes the development team's life easier:

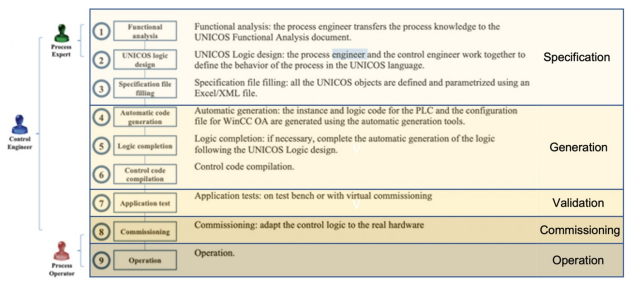


Figure 4: UCPC development methodology.

a) Specification: To produce the initial specification (step 1, 2, 3 in Fig. 4) the process experts use a common set of basic documentation elaborated during the design phase of the project:

- The electrical schemas, the Process & Instrumentation Diagrams (P&ID) and the instruments lists together with their associated data to validate the conformity with the cryogenic group prescription for construction and protection principles;
- The P&ID to elaborate the Process Flow Diagram (PFD), to identify the functions and the completeness of the proposed instrumentation to ensure the requested duties. They permit also to validate if the tag names proposed are in conformity with CERN required standards;
- The instruments lists to identify the conformity with the prescriptions of the components choices and to validate the availability of the necessary process data.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

From these documents it is possible, in collaboration with a control expert, to prepare either manually or automatically:

- The devices description specifications as XML format sheets to describe all devices to be used within an application;
- The functional and logic specifications based on text templates used to code the needed PLC function placeholders in a standardised way using predefined templates;
- The supervision configuration data and synoptics that can be either drawn manually or generated using standardised templates.

A quality testing of the data validity can then be conducted to maintain the maximum reproducibility and to reduce the commissioning time (comparison of the parameters stored in the databases together with the parameters extracted from the production system). Today the non LHC cryogenic systems and the LHC cryoplants are manually developed whereas the cryogenic tunnel applications are automatically generated.

b) Generation: When generating the applications (step 4, 5, 6 in Fig. 4), the XML inputs files describing the devices as well as their properties and the python templates for PLC codes and SCADA panel, are processed and combined by the UNICOS Application Builder (UAB). Its internal plugins follow semantic rules to produce the application files to be loaded directly in PLCs and SCADA. If at this stage templates are missing, the control expert must either manually complete the control logic placeholder in the PLC or draw the process synoptics with the SCADA drag & drop tool.

c) Application Validation or Virtual Commissioning (step 7 in Fig. 4) can be performed to validate the implemented logic:

- Application validation : A mirror PLC with automated or semimanual features, to control the different thresholds for the interlocks logical transitions, is used [4];
- Virtual commissioning : A cryogenic model using a dynamic simulation model based on EcosimPro® and a Cryogenic library developed at CERN is exploited to validate the control loops through the UCPC decomposition, allowing a complete virtual commissioning of the system.

d) Final Cryogenic Commissioning (step 8 in Fig. 4): After the completion of the previous validation tests, process experts can proceed to the actual commissioning to adapt the application to the process.

e) Operation: In this last step the operators use the SCADA User Interface (UI) to take over normal operation. In case of failure, they can obtain detailed information about the entire process and controlled devices. For instance, they have the access to diagnostic panels to investigate the origin of interlocks (mechanical, electrical, hardware, control software, etc.) in order to take quick corrective actions to overcome problems and reduce potential downtime.

Automation of the Application Generation

The complexity of the cryogenic control system and the frequent updates carried out for optimisation or regular maintenance have made the follow-up of the control software complicated and time-consuming with high demand for complex Information Technology (IT) tools. To elaborate automatically the UAB inputs files and improve the quality of the cryogenic control applications, an adapted generator using the design principle of the "Continuous Integration" model has been developed [5]. This generation process uses a shared project control repository, with the XML devices fill, the process logics and generic panels implemented in Python templates easing the replication and automates all the steps to build new projects including the new code testing. Hence, since it's implementation the applications controlling the LHC tunnel, have been regularly updated, with excellent results in terms of efficiency and reliability such as the reduction of the necessary time for the code review. The different languages semantic and the lack of inter-operability between PLC brands and the integration of PLC-SCADA with common protocols were the main limitations of industrial control systems. The use of UCPC solved a large part of these limitations. Applications generation for the various brand of PLC are homogeneous since they are based on a common control model. The PLC-SCADA communication configuration is automated and the communication from PLC to SCADA uses a Time Stamped Push Protocol (TSPP) adapted to the PLC brand. All the application generation processes have been automated and implemented.

CRYOGENIC CONTROL SYSTEM EVOLUTION IN THE NEAR FUTURE

Beside the process control applications needs continuos to be update in both the supervision and control levels to follow the operation requested evolution (synoptic evolution, new trends, new alarms, etc.), to implement the logic modifications related to the hardware protection evolution,to follow the evolution of the requested logic & presentation and to develop additional process operation features. The goal is to facilitate the operators duties to improve the application software productivity, to reduce the critically of several components improving the hardware availability, and to simplify the preventive maintenance reducing the costs [1].

Supervision Level Evolution to Facilitate the Operator's Duties

A new version of WinCC-OA® and the UCPC SCADA framework package will be introduced [6], in addition to a new archiving technology compatible with the NXCALS storage and a new alarm list with additional functionalities. The new version offers a large integration with ML layer

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

information tools from each device (e.g., CMMS data, Control Logic, Electrical Drawing) that will be extended in the coming years.

Improving Application Software Productivity

A recent extraction of the control devices has shown that around 280000 UCPC devices are already deployed and used for the various cryogenic controls systems at CERN. This gives a total of more than 13 million fields to be maintained. Since in the specification phase of UCPC, technical documents contain all the technical information needed to develop the cryogenic control system applications. Therefore, the needed evolution is to make the complete set of technical data contained in those documents available. Thereby, the various stakeholders will be able to access all necessary information to develop, operate or maintain the cryogenic system [7]. Besides this work, a Control Specification Generator including a new database as well as a set of generic software components to extract the information to produce the XML input files for UAB will have to be developed [7, 8]. An official request to the CERN Industrial Control group to prepare the generation database and develop the tool has been filled, accepted and noted in the Road-Map. However, the development is presently on hold.

Other Generators

With a similar approach, an evolution of UAB and the development of adapted python templates and rules should be generated for new projects or renovation of existing ones:

- EcosimPro® & CryoLib, cryogenic simulation models;
- Cryogenic Maintenance plan for MMS (procedure, frequency, checklist, etc.);
- Data driven schematics;
- Intelligent electrical schemas;
- Intelligent mechanicals P&ID

Improving the Control and Instrumentation Availability

To enhance the availability of the control and instrumentation hardware there is a need to eradicate the root causes of failures, to reduce the fault diagnostic time and to minimize the equipment variability, eliminating at the same time unnecessary equipment to limit the annual maintenance costs. This goal will be achieved by using industrial fieldbus, standard cabinets and removing outdated plugs and installing push terminals:

- The PROFIBUS® communication will be updated with the migration to Ethernet-based technology for PROFIBUS® DP long distance Network, and the replacement of PROFIBUS® PA I/O board to introduce additional redundancy.
- The 400 V distribution design will be modified to simplify the distribution and protection. Main maintenance switches will be installed on the cabinets to facilitate and accelerate the recovery in case of failure.
- The 24 V devices will be supplied with a bus to enhance the flexibility, ease the diagnostic and improve the availability in case of a component failure.

Radiation Tolerance Consolidation

The LHC cryogenic system has operate during 2 physics Runs without radiation induced failures. However, in the coming years, with the increment of the luminosity in the frame of the HL-LHC, this aspect must be considered to avoid failures that will lead to unplanned downtime. Several solutions have been explored (i.e. transfer the cryoplants to a surface building or to existing underground protected locations or creation of a new service caverns) The most suitable solution seems to shield by a wall between the source of the radiation and the sensitive equipment, in order to reduce the numbers of events [9]. The problem of this installation is the crowded environment and the thickness of the wall to achieve the right reduction of radiations as an attenuation by a factor of 5 is the goal to achieve when the luminosity will rises to $50\,fb^{-1}/y$.

CRYOGENIC CONTROL SYSTEM TOMORROW

This chapter is based on real concepts studied and, for a part of them, already in practice in the contemporary control community. First of all, the groups do agree with many assertions presented in the [10] (chapter 5.10) and in particular with those going in a direction, particularly true and necessary for the control systems, to make the choice of homogeneous solutions for the various control systems (infrastructures, accelerators & detectors) and to open large cooperation with industrial, research and institutional partners. Strong partnership with industrial providers can be a key to the success of a project. Indeed, specific requirements that drastically differ from those of industry (scale of the applications, integration with external or exotic systems, flow control and ultra-fast real time constraint) have to be offered. For the evolution of the control systems, the cryogenic group preference, is to remain as compatible as possible with its present implementation and follow the state-of-the-art. Then, many components or services will have to be developed and maintained by the appropriate entities: IT department, controls groups. The plan is to collaborate as much as possible to develop the most adapted solutions.

Architecture Evolution

Today the buzzwords for the future of industrial automation are Internet of Things (IoT) and Service Oriented Architecture (SOA). Internet of things applies to the devices/components found in the control system [11]. With IoT the control devices are no longer at the lowest level of the control pyramid but they are the node of interactions with several services across the levels such as presented in Fig. 5:

- Service-oriented automation : The application domain is the source of knowledge. Several techniques can be used to describe the available information (e.g. the use

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

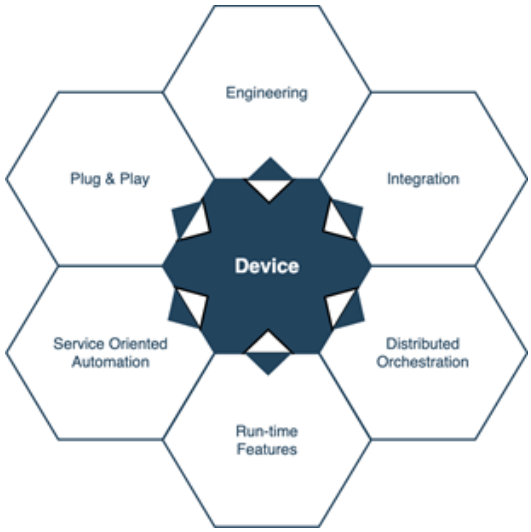


Figure 5: IoT Device model.

of semantics, ontology), besides the intrinsic capabilities provided by web service technologies to capture them in an automated way;

- Plug & play is the ability to connect devices to a network with minimal need of configuration as they are able to integrate themselves and will be ready to work. It uses dynamic discovery, dynamic configuration, uniform description of the device functions and use of standard interfaces;
- Engineering represents the methodologies and tools to make the controlled system run and behave in accordance to what is expected. Services engineering covers every aspect from the initial analysis phase to the operation & maintenance phase;
- Integration covers not only the integration of devices with others, but also their interaction with other levels (e.g., entire production systems, Business);
- Distributed orchestration, since the coordination of activities is not viewed in a central manner, the collaboration between separated units need the use of complex interactions such as orchestration and composition of control to achieve the global objectives.
- Run-time features: When the system is running, there is a continuous monitoring of several useful features to automatize the process to improve the maintenance and to take actions as soon as possible failures occur. The distinctive features are considered to be re-configurable, adaptable, intelligence, auto-sustainable.

In Service Oriented Architecture (Fig. 6), the hierarchical communication of the classical layered architecture, is replaced by a flat exchange of information. The services used in the process can be easily modified with a cross layer integration to improve the inter-operability. The embedded services incorporated in the system operate independently [12]. For this reason, adding or modifying new services and processes will become easier. As result, the process is no longer constrained in a specific platform, but it can be considered as a component of a wider process and therefore reused or modified.

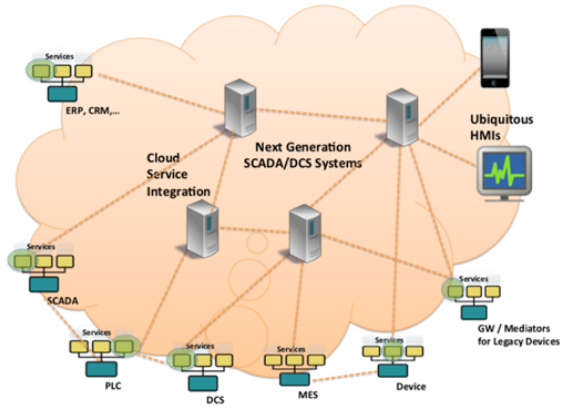


Figure 6: Next generation of industrial control system, service based, and flat information driven interaction [12].

Evolution of the Control Architecture

The transition to SOA technology has already been implemented in the highest layer of the enterprise system (CL) for many years. It is also used for the integration between CL and ML with solutions available on the market. The downward spread to SL, CL and IL is progressing today thanks to some initiatives such in Industry 4.0 and existing industrial protocols covering a large part of the necessary services such as OPC-UA (Fig. 7).

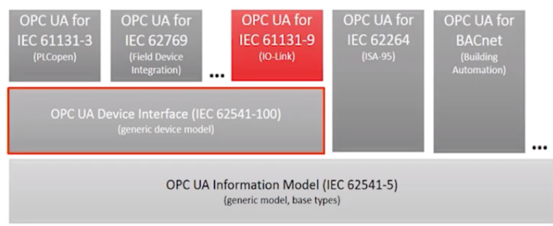


Figure 7: OPC-UA information model that can already be used as generic middleware for slow control systems.

To develop a SOA based control system, the Reference Architectural Model Industries 4.0 (RAMI 4.0) presented in Fig. 8 could be considered. The hierarchy level axis may integrate the accelerators and detectors equipment (field devices and control devices) with the system levels (connected world, enterprise, workcentre, station). Consequently, services, interfaces and integration guidelines shall be used to allow the aforementioned components to interact with the different functional levels (business, functional, information, communication, integration and asset). Ultimately, the life cycle management framework shall be used by the development process to span from concept over requirements, design, implementation/procurement, transition to operation, maintenance and retirement.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

This model is an ideal reference to establish the control and data acquisition architecture for an entire project, limiting internal project developments to the only parts that cannot be covered by existing products, services, standards and guidelines.

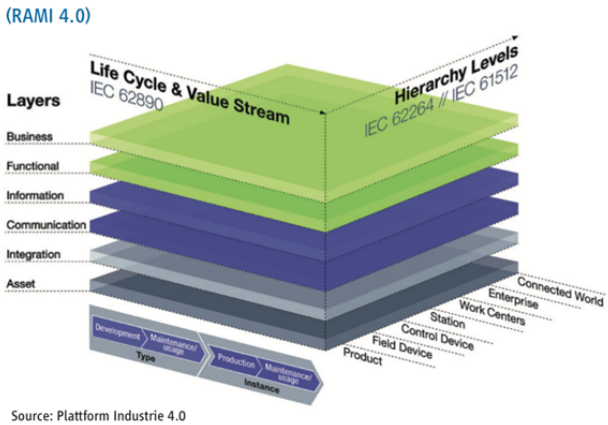


Figure 8: Reference Architectural Model Industrie 4.0.

Migration from the UCPC Cryogenic Control System

To adapt the cryogenic control layer to the SOA architecture, the following challenges must be faced [13]:

- One of the strengths of the cryogenic control architecture is the tight link between the HMI and the control execution with the operator's intervention restricted to the chosen devices. Hence, further break down of the hierarchy shall not affect the functional integration. In addition, the choice of an adapted middleware such as OPC-UA will offer additional features.
- Group the devices: Within a given system, it must be determined which device can be migrated to SOA as devices and which devices shall be grouped together and then migrated to SOA as a group.
- For process control execution the present architecture allows an excellent integration between devices if the expected behaviour by the process expert has been implemented. However, in case of unexpected event, adapted behaviour will have to be automatically initiated from the lowest level possible and propagated with automation services (such as OPC-UA for IEC61131.3).
- Preserve real-time control: The real time control execution, which, in the legacy system, is secured in the controllers, must be preserved [14]: As example, a subsystem using feedback and regulation might require legacy interfaces due to real-time needs. For such a group of devices the SOA interface will be implemented using a Mediator for the entire group and this part of the system will be handled as a black-box.

Hardware Evolution from High Luminosity LHC to Future Circular Collider (FCC)

Hardware for the cryogenic control systems should be able to support the software evolution towards IoT and SOA. At the low level of the pyramid, PLCs were the best choice for systems such as cryogenics, cooling, ventilation, vacuum, etc. because they are simple, with an efficient programming language, high level of availability and reliability. However, they lack the capabilities to run parallel equation solving algorithms or high level programming languages necessary to implement the IoT and SOA complex capabilities such as dynamic discovery and configuration adaption with process conditions typical of IoT devices or the interconnection of services (configuration, maintenance data exchange, etc.) necessary for SOA. Even though the features just mentioned can be obtained thanks the LHC Front End Computer (FEC), their reliability is not as high as that of the PLC. New solutions, such as the Beckoff Twincat PLC, must be investigated to obtain a product compatible with the new concepts with a very high reliability. Today the low-level devices used in cryogenic systems do not integrate complex capabilities such as dynamic discovery and configuration adaption with process conditions typical of IoT devices. However, some of IoT compatible devices already exist (such as IO-link compatible devices). In addition, Industrial Control Middleware such as OPC-UA proposes a mediator for the interconnection of services (configuration, maintenance data exchange, etc.). For the communication to the process interface at the lowest level of the pyramid, the main concerns are related to the electronic control components located inside the accelerator tunnel, due to the harsh conditions. For the FCC, a FLUKA model of the collider arc has been created. The results obtained, show increment factors of about 500 time compared to the LHC for the stochastic and the cumulative effect respectively. But the same model shows that shielded alcoves radiation levels are such that they can host electronics devices with a proper lifetime and with stochastic effects reduced to an acceptable rate. However, some electronic components will have to be implemented close to the accelerator. As a consequence, the major concerns of the cryogenic control system for this level, will be the data acquisition and the powering means between the sensors/actuators located in the vicinity of the beams and the processing/driving activities that will be in the alcoves. The FCC design report proposes to study the hardening of a cabled Ethernet based solution by applying the recommended "The FCC-hh Radiation Hardening Availability (RHA)", founded on a full-availability approach based on moving the processing tasks away from the equipment under control, introducing functionalities for self-diagnosis of failure, online hot swapping, remote control and remote handling.

A potential solution could be to use a CERN custom-made module, the Distributed I/O Tier (DI/OT). In this solution, a generic low cost customisable and reusable communication intermodule is deployed (see Fig. 9) to achieve the needed for radiation-exposed and radiation-free areas. This platform

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

is based on compact PCI serial communication. It allows the designers to use standard elements and standard voltages. The hardware kit includes different interchangeable modules ("1" for radiation-exposed ares and "2" for the free radiation areas) [15]:

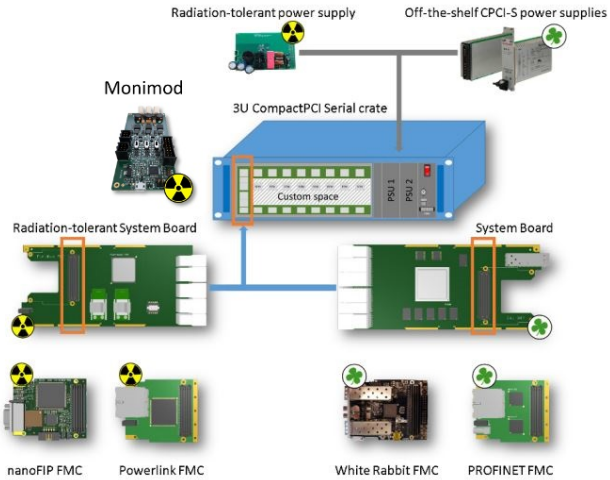


Figure 9: Distributed I/O Tier hardware [15].

- Power supply:
  - "1" RaToPUS: hardened 100 W AC/DC switched mode power supply ;
  - "2" Off-the-shelf CPCI-S power supply: 300 W switched-mode power supply.
- Monitoring module Monimod to monitor simple functionalities such as voltage, current, temperature or the control of an additional fan tray temperature and speed in radiation exposed area ;
- FPGA based system board with replaceable fieldbus communication mezzanines to allow different communication protocols:
  - "1" WorldFIP and Ethernet-POWERLINK
  - "2" White Rabbit and PROFINET

In addition, considering the burden (resources, planning and costs) faced by the LHC cabling campaigns for each LS, wireless communication and power supply from the alcoves to the sensors/actuators, would be an excellent solution. It has been proposed in the FCC week 2015 [16] and since then, two initiatives WADAPT [17] and The "Smart Diagnostics" from the FuSuMaTech project [18] have presented the same type of proposal based on 60 GHz wireless CMOS Chips including IoT features for the second one

Responsibility Sharing for Future Large Projects

For future projects, the sharing of responsibilities must be organized at the laboratory level. Hence, the CERN control community will be able to address the challenges proposed by the evolution of technologies. To achieve this goal the project need to profit of the experience at CERN labs or within successful control collaborations and last but not least to setup large collaboration frameworks for hardware and software.

CONCLUSION

The LHC cryogenic control is the result of years of experience with this highly demanding process that can induce very long accelerator downtime in case of failure. To achieve the present availability, an automated and continuously updated cryogenic process was considered and implemented thanks to a good synergy between the cryogenic group and the accelerator controls groups.

Today, the ongoing CERN HL-LHC upgrade project requests the cryogenic system to be more reliable, versatile, and still easy to implement. These goals will be achieved with the collaboration of a large part of CERN control community spread in at least 5 CERN departments.

This evolution must start with a serious consciousness of the experience within successful control collaborations (e.g. Tango, EPICS), the integration of versatile procedures and protocols in parallel with several open and active collaboration frameworks (for hardware and software parts). It will be crucial to set the right balance between open solutions, solutions based on open standards, off the shelf's components and proprietary solutions. Special attention must be given to keep strategical domains under control, avoiding vendor lock-in issues and allowing a fair knowledge transfer between partners. This new architecture shall cover all the levels of the ancient control pyramid and offers all necessary services proposed by the SOA approach.

To integrate the presented technological breakthrough and not miss any technological rupture the CERN (infrastructures, accelerators systems, experimental physic, and the information technology) control community shall be included together with CERN research partners and partnership with the industry to federate the efforts. Our recommendation for the needed collaboration frameworks is to start as soon as possible (I.e., the hardened hardware components) with a strong management support.

REFERENCES

[1] M. Pezzetti, "Control of large helium cryogenic systems: a case study on CERN LHC," *EPJ Techniques and Instrumentation*, vol. 8, no. 1, pp. 1–20, 2021, doi:10.1140/epjti/s40485-021-00063-w

[2] J. Casas-Cubillos, P. Gomes, P. Gayet, F. Varas, C. Sicard, and M. Pezzetti, "Application of object-based industrial controls for cryogenics," in *Proc. EPAC'02*, Paris, France, 2002, paper MOPDO030, pp. 2013–2015, https://jacow.org/e02/PAPERS/MOPDO030.pdf

[3] P. Gayet and R. Barillère, "Unicos a framework to build industry like control systems: Principles & methodology," in *Proc. ICALEPCS'05*, Geneva, Switzerland, 2005, paper I3_002, https://jacow.org/ica05/proceedings/pdf/I3_002.pdf

[4] C. Fluder *et al.*, "Improved software production for the lhc tunnel cryogenics control system," *Physics Procedia*, vol. 67, pp. 1134–1140, 2015, Proceedings of the 25th International Cryogenic Engineering Conference and International Cryogenic Materials Conference 2014, doi:10.1016/j.phpro.2015.06.176

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[5] T. Barbe and M. Pezzetti, "An innovative approach for the design of cryogenic electrical and process control systems at cern: The cryogenic continuous integration project," unpublished.

[6] P. Golonka and F. Varela, "Consolidation and Redesign of CERN Industrial Controls Frameworks," in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 963–970, doi:10.18429/JACoW-ICALEPCS2019-WEDPL04

[7] E. Estévez, M. Marcos, and D. Orive, "Automatic generation of plc automation projects from component-based models," English, *The International Journal of Advanced Manufacturing Technology*, vol. 35, no. 5-6, pp. 527–540, 2007, doi:10.1007/s00170-007-1127-4

[8] L. Serio *et al.*, "CERN experience and strategy for the maintenance of cryogenic plants and distribution systems," *IOP Conference Series: Materials Science and Engineering*, vol. 101, p. 012 140, 2015, doi:10.1088/1757-899x/101/1/012140

[9] K. Buffet, F. Butin, and F. Cerutti, *A shielding wall in UX85 for LHC run 5*, 2020, https://indico.cern.ch/event/946017/contributions/4047622/attachments/2114676/3557740/LHCb_wall_proposal_CDR_v0.1.pdf (accessed: 25.08.2021).

[10] *Future Circular Collider Conceptual Design Report*, doi:10.1140/epjst/e2019-900087-0

[11] J. Mendes, "Engineering framework for service-oriented automation systems," Ph.D. dissertation, Faculdade de Engenharia da Universidade do Porto, 2011, https://repositorio-aberto.up.pt/bitstream/10216/63301/1/000150082.pdf

[12] S. Karnouskos and A. W. Colombo, "Architecting the next generation of service-based scada/dcs system of systems," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, 2011, pp. 359–364, doi:10.1109/IECON.2011.6119279

[13] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, and T. Bangemann, "Migration of industrial process control systems into service oriented architecture," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 5786–5792, doi:10.1109/IECON.2012.6389039

[14] G. Chen and B. An, "Towards a Time-Constrained Service-Oriented Architecture for Automation and Control in Large-Scale Dynamic Systems," in *Proc.ICALEPCS'17*, Barcelona, Spain, 2018, pp. 760–764, doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

[15] G. Daniluk, C. Gentsos, E. Gousiou, L. Patnaik, and M. Rizzi, "Low-Cost Modular Platform for Custom Electronics in Radiation-Exposed and Radiation-Free Areas at CERN," in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, pp. 671–677, doi:10.18429/JACoW-ICALEPCS2019-TUAPP03

[16] P. Gayet, "FCC Control Systems Concepts: Why it's not too early to speak about it," in *2nd FCC Workshop*, 2015, https://indico.cern.ch/event/340703/contributions/802257/attachments/668832/919357/control_systems_for_FCC.pdf

[17] H. K. Soltveit, "Multi-Gigabit Wireless Data Transfer for High Energy Physics Applications," in *Proceedings of The European Physical Society Conference on High Energy Physics — PoS(EPS-HEP2017)*, vol. 314, 2017, p. 520, doi:10.22323/1.314.0520

[18] I. Aziz, D. Dancila, and R. Brenner, "Testing electronic components for low temperatures wireless communication link - LTCL," https://phase1.attract-eu.com/wp-content/uploads/2019/05/LTCL.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

THE CONTROL SYSTEM OF THE NEW SMALL WHEEL ELECTRONICS FOR THE ATLAS EXPERIMENT

P. Tzanis*,[1,2], on behalf of the ATLAS Collaboration
[1] National Technical University of Athens, Athens, Greece
[2] Brookhaven National Laboratory, Upton, NY, USA

Abstract

The present ATLAS Small Wheel Muon detector will be replaced with a New Small Wheel (NSW) detector in order to cope up with the future LHC runs of high luminosity. One crucial part of the integration procedure concerns the validation of the electronics for a system with more than 2.1 M electronic channels. The readout chain is based on optical link technology connecting the back-end to the front-end electronics via the FELIX, which is a newly developed system that will serve as the next generation readout driver for ATLAS. For the configuration, calibration and monitoring path the various electronics boards are supplied with the GBT-SCA ASIC and its purpose is to distribute control and monitoring signals to the electronics. Due to its complexity, NSW electronics requires the development of a sophisticated Control System. The use of such a system is necessary to allow the electronics to function consistently, safely and as a seamless interface to all sub-detectors and the technical infrastructure of the experiment. The central system handles the transition between the probe's possible operating states while ensuring continuous monitoring and archiving of the system's operating parameters.

NEW SMALL WHEEL

In order to efficiently handle the increased luminosity that will be provided by the High-Luminosity LHC (HL-LHC), the first station of the ATLAS [1] muon end-cap system (Small Wheel, SW) will need to be replaced. The New Small Wheel (NSW) [2] will have to operate in a high background radiation region (up to 22 kHz/cm²) while reconstructing muon tracks with high precision as well as providing information for the Level-1 trigger. The detector technologies to be used come from the family of gaseous detectors, the first is called small-strip Thin Gap Chambers (sTGCs), and the second comes from the category of micro-pattern gas detectors and is named Micromesh Gaseous Structure (Micromegas (MM)) [3] . The new experimental layout will consist of 16 detection layers in total and 8 layers per detection technology (8 layers sTGC and 8 layers Micromegas), as shown in Fig. 1. The sTGC detectors are designed to provide fast trigger and high precision muon tracking under the HL-LHC conditions. On the other hand, Micromegas detectors have a small conversion region (5 mm) and fine strip pitch (0.45 mm) resulting in excellent spatial resolution and are primarily used for precise tracking.

* polyneikis.tzanis@cern.ch
[0] Copyright [2018] CERN for the benefit of the [ATLAS Collaboration]. CC-BY-4.0 license.

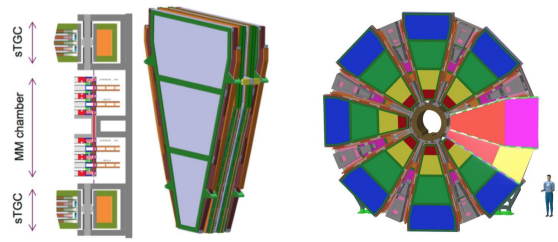


Figure 1: A graphic representation of the NSW sector (left) which consists of 8 layers of Micromegas in the inner part and sandwiched by 4+4 layers of sTGC detectors in the outer parts and view of the NSW (right) with 16 sectors in total. [4]

ELECTRONICS OVERVIEW

The NSW electronics for the trigger and Data Acquisition (TDAQ) path of both detectors is divided into two major categories, on-detector and off-detector electronics, as shown in Fig. 2. On the left-hand side, the front-end boards that

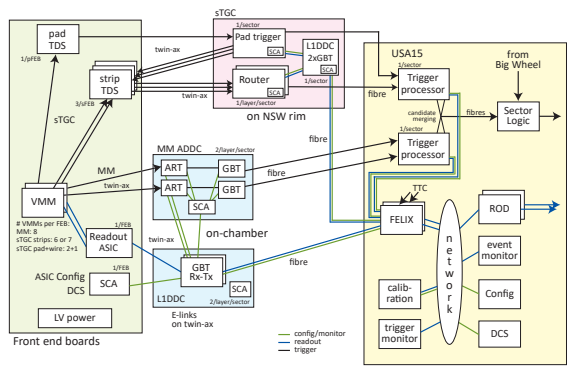


Figure 2: Overview of the NSW electronics scheme. The front-end detector boards are depicted on the left (for MM and sTGC), the data-driver cards (L1DDC, ADDC) in the middle while the back-end electronics can be seen on the right. [4]

are attached to the chambers bear VMMs, SCAs(Slow Control Adapter) and ROCs(Read Out Controller), while for the sTGCs they also host TDS chips. The ROC aggregates L1 data from many VMMs [5] and sends them to (Front End LInk eXchange (FELIX) [6] via the GBTx [7]. FELIX also sends trigger signals to the front-end electronics via the ATLAS TTC system. It also sends tracking data from the ROC to the swROD (here depicted to send the data fragments to the HLT), and communicates with the Detector Control System (DCS) [8] for slow control purposes. The Micromegas trigger data are collected from many VMMs by

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the ART ASIC, which sends them towards the Micromegas trigger processor, alongside geographical and timestamp information. The TDS chips on the other hand, handle the sTGC trigger primitives, and alongside the Pad Trigger and Router boards, are part of the more complex sTGC trigger chain, which has the associated trigger processor FPGA on the back-end as a final destination of its data-flow. Both FPGAs create muon candidates that are then transmitted to the SL. The NSW on-detector electronics (Front-End boards (MMFE8), Level-1 Data Driver Card (L1DDC), ART (Address in Real Time) Data Driver Card (ADDC)) will be placed inside the cavern (detector area with radiation and magnetic fields) and consists of custom-made boards mainly using radiation-tolerant Application Specific Integrated Circuits (ASICs). The communication between these boards will be established with the use of mini Serial Attached Small Computer System Interface (SCSI) cables. The off-detector electronics FELIX, trigger processor, sector logic and services running on commercial server computers like Read Out Drivers (ROD), Detector Control System (DCS), event monitoring, configuration, trigger monitor and calibration) will be placed outside the cavern in an area that is called USA15.

FELIX

The keystone of the ATLAS DAQ system will be the FELIX which is an FPGA-based system housed by a commercial server. FELIX will essentially be a bridge between the front-end electronics of all ATLAS detector subsystems, and their corresponding back-end components, which will mostly be software-based, as shown in Fig. 3. Situated in

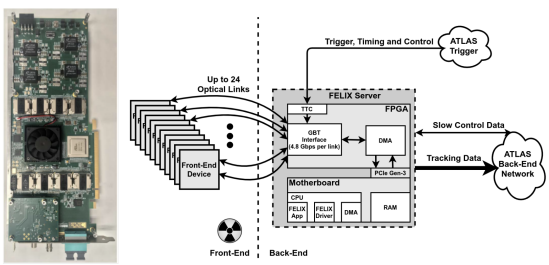


Figure 3: Left: The FELIX BNL712 FPGA board, which the NSW will use for its needs. It features a Xilinx® Kintex Ultrascale XCKU115 FPGA, a PCIe connector to interface with the CPU and the back-end network, and an optical coupler supporting up to 24 links. A dedicated mezzanine board receives the reference clock and the trigger information from the ATLAS TTC system. Right: FELIX and its relationship with other parts of the DAQ system.

the USA15, FELIX connects to the front-end electronics of the ATLAS cavern via optical links, or GBT links, each one of which is running at 4.8 Gb/s. For the NSW case, FELIX will interface with the front-end nodes over 24 optical links. These links carry the GBT frame, which is 84-bit wide. The Giga-Bit Transceiver (GBT) protocol is a transmission scheme that involves radiation-tolerant ASICs that are capable of handling the large amounts of data of high energy physics experiments.

GBT-SCA

The GBT-SCA ASIC (Giga-Bit Transceiver - Slow Control Adapter) [9] is an integrated circuit built in a commercial 130 nm CMOS technology and is the part of the GBT chipset which purpose is to distribute control and monitoring signals to the front-end electronics embedded in the detectors. It connects to a dedicated electrical port on the GBTx ASICs through an 80 Mbps dual redundant bidirectional data-link; namely the e-links. In order to meet the requirements of different front-end ASIC in various experiments, the SCA provides a number of user-configurable electrical interface ports, able to perform concurrent data transfer operations. The user interface ports are: 1 SPI master, 16 independent I2C masters, 1 JTAG master and 32 general-purpose IO signals with individual programmable direction and interrupt generation functionality. It also includes 31 analog inputs multiplexed to a 12 bit ADC featuring offset calibration and gain correction as well as four analog output ports controlled by four independent 8-bits DACs. An illustration of the different blocks are displayed on Fig. 4.

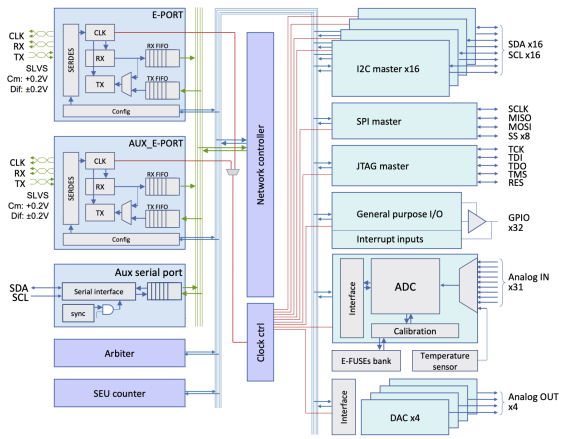


Figure 4: Overview of the GBT-SCA block diagram. [9]

SCA OPC-UA SERVER

For the NSW project alone, ≈ 6400 SCAs will be used to configure and monitor various ASICs that are part of the general DAQ scheme. All of these SCAs communicate with the back-end (i.e. FELIX) via the GBTx, which implements the GBT protocol in its logic, while FELIX implements the GBT-FPGA core in order to communicate with the aforementioned package. The SCA-to-GBTx interface is the E-link, a serial differential line running at 80 Mb/s over HDLC encoding. The protocol defined by the SCA's requirements is request/respond that is, for every packet received by the SCA, the back-end that originally transmitted the packet awaits for an associated reply by the SCA. In this SCA-GBTx-FELIX communication chain, the last two components can be viewed as data mediators, so there is one piece missing: the back-end logic that actually

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

builds the packets-to-be-transmitted to the SCA, and handles the inbound traffic from the ASIC. This is a software suite, which is a dedicated Open Communications Platform Unified Automation (OPC-UA) server. Overview of the on/off-detector electronics and the DCS, Configuration and Calibration path via the SCA, the FELIX and the SCA OPC UA Server is illustrated in Fig. 5. A common readout path and a separate trigger path are developed for each detector technology. The communication chain starts in the control room with an OPC-UA client, which is the first step into our detector control system. Configuration and monitoring data is requested and sent from here to the OPC-UA server and next to the FELIX PCs, which lie in the underground service area. Then, through radiation-hard fibers, we reach the GBTx chips, which implement the optical links, to finally arrive to the SCA.

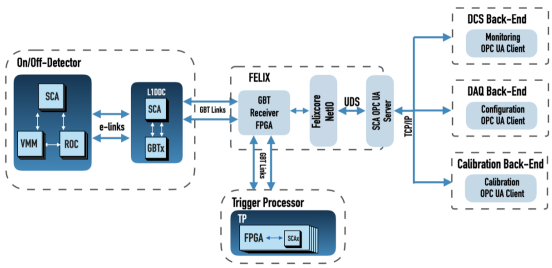


Figure 5: Representation of the on/off-detector electronics and the DCS, Configuration and Calibration path via the SCA, the FELIX and the SCA OPC UA Server [4]

ATLAS DETECTOR CONTROL SYSTEM

The ATLAS DCS has the task to permit coherent and safe operation of ATLAS and to serve as a homogeneous interface to all sub-detectors and the technical infrastructure of the experiment. The DCS must bring the detector into any desired operational state, continuously monitor and archive the operational parameters, signal any abnormal behavior. The DCS was designed and implemented within the frame of the Joint Controls Project (JCOP), a collaboration of the CERN controls group and DCS teams of the LHC experiments. Standards for DCS hardware and software were established together with implementation guidelines both, commonly for [10] and specifically for ATLAS. It combines common standards for the use of DCS hardware based on SCADA system Siemens, WinCC Open Architecture, also known as PVSS with its older name, where it serves as the basis for all DCS applications. Figure 6 depicts the architecture of DCS which can be divided into Front-End (FE) equipment and a system Back-End (BE).

FE includes equipment DCS, including customized electronic systems and related services such as high voltage power supplies and cooling circuits. The BE system uses the software WinCC, integrating front-end control systems into the framework components to facilitate the integration of standard hardware devices and the implementation of homogeneous control applications. The two ends of DCS

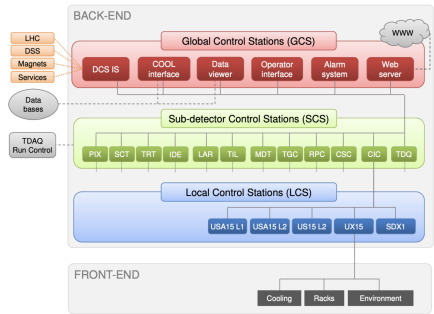


Figure 6: Graphical representation of the DCS architecture of the experiment ATLAS divided into 3 levels: GCS, SCS and LCS. [8]

communicate mainly through the industrial protocol bus CAN, while the communication standard OPC is used as a communication software protocol. The system BE is organized hierarchically on three levels, the Local Control Stations (Local Control Stations (LCS), the Sub-detector Control Stations (SCS)) and the Central Control Stations (Global Control Stations (GCS)). In total, BE consists of more than a hundred computer stations connected to a distributed system. Communication between subsystems of the system is handled by WinCC via a local area network. A distributed finite state machine, Finite State Machine (FSM), represents the complete hierarchy of the BE system as it integrates more than 10 million data elements into a single tree structure and ensures proper operation and efficient handling errors in each operating layer. The most important element of this system is the structure data point (DP), which plays the role of the global variable network. Each element of this structure has a unique name and configurability, and special DPs are used to read data from the hardware components updated by the interface OPC client-server communication.

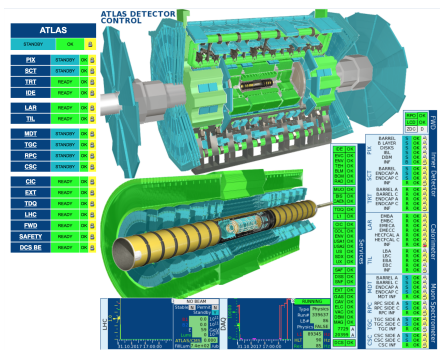


Figure 7: Operator interface (FSM Screen) showing the detector in STANDBY configuration during LHC ramp-up. The top hierarchy level object is shown together with its children objects (top left) and the associated main panel (bottom right). The UI allows the navigation to any FSM object and associated panel within the whole ATLAS DCS hierarchy. On the top right, a list of objects with non-OK Status allow shortcut navigation to problems. [8]

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Each node FSM has a unique name based on the subsystem name and its functionality following the conventions of ATLAS DCS and the state in which they are specified by a corresponding internal DP. The type of object FSM which defines the basic functionality of the node and its components, depends on the functional purpose and position of the element in the DCS architecture hierarchy. The main graphical user interface of ATLAS DCS with all subsystems integrated into a hierarchical structure FSM is illustrated in Fig. 7. The FSM is based on a strict hierarchical structure that constitutes parent-child relationships, where in tree construction commands are passed from parents to children, and situations from children to parents. This way, when some action is required on all children it is extremely efficient to command a higher node and correspondingly the status of the higher node summarizes the status of all nodes of any generation. All nodes are in a predefined state and only accept predefined commands as defined in the FSM type to which they belong. The DCS is operated from two primary, remotely accessible user interfaces – the FSM Screen for operation of the detector Finite State Machine hierarchy (see Fig. 7) and the Alarm Screen for alarm recognition and acknowledgment. Static status monitoring is provided by web pages on a dedicated web server allowing to quickly visualize all high level FSM user interface panels world-wide and without additional load of BE control stations.

NSW DETECTOR CONTROL SYSTEM

Due to its complexity and long-term operation, the NSW requires the development of a sophisticated Detector Control System (DCS). The use of such a system is necessary to allow the detector to function consistently and safely as well as to function as a seamless interface to all sub-detectors and the technical infrastructure of the experiment.

Architecture

The NSW DCS architecture and its integration with the ATLAS DCS have been finalized and projects will closely follow the existing look, feel and command structure of Muon DCS, to facilitate the shifter and expert operations. The current plan is to have 2 new sub-detectors, MMG (Micromegas) and STG (sTGC). The top node of both MMG and STG will propagate its state and receive commands from the ATLAS overall DCS. An overview of ATLAS MUO DCS structure with NSW DCS integration and brief structure of the sub-detector node structure is displayed in Fig. 8. The overview of the NSW structure and project/server allocation is displayed in Fig. 9.

ELECTRONICS CONTROL STATION

System Setup

The electronics system setup will consist of 28 FELIX servers, 12 for MMG and 16 for STG accordingly. The FELIX-SCA OPC UA Server granularity will be 1-to-1 meaning that we will have 1 SCA OPC UA Server per sub-detector sector. An example is illustrated on Fig. 10.

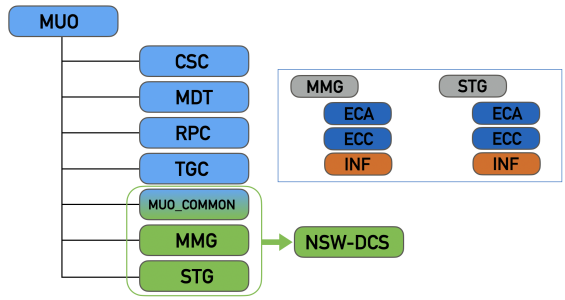


Figure 8: The overview of the ATLAS MUO DCS structure with NSW DCS integration and brief structure of the sub-detector node structure.

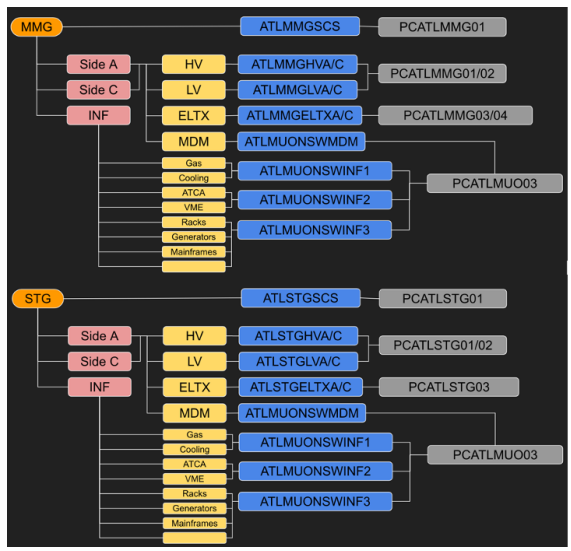


Figure 9: The overview of the NSW structure and project/server allocation.

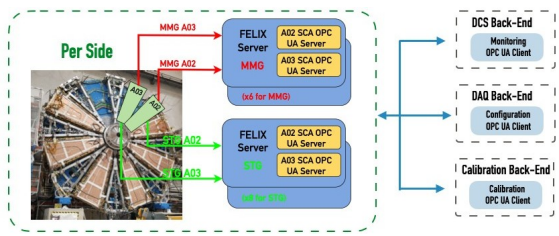


Figure 10: An illustration of the FELIX-SCA OPC UA Server granularity mapping.

For the system operation, automated sequence has to be settled up on the FELIX machine in order to perform the GBT e-link configuration, the FELIX API initialisation and the SCA OPC UA Server automatically in the FELIX machine boot.

Configuration

The NSW is a fully autonomous trigger and tracking detector system, adequately supported by an advanced electronics scheme and ready to handle the challenges of increased instantaneous luminosity at the High Luminosity LHC. It

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

includes more than 60k front-end ASICs and a few tens of FPGAs, which need to be configured before every run of the experiment. This process needs to be efficient and quick, since it needs to happen a few times per day. The main ASIC to be configured is the VMM, the front-end signal pre-processing chip that can readout 64 channels. A few kbytes of configuration data include thresholds for each channel, but also global registers to define the gain, time-to-amplitude conversion and many others. For this procedure we need to use the SCA's SPI master to communicate with the 8 SCA's SPI slaves in the VMMs. Also, the SCA's GPIO interface is required, to act as an enable signal. Moreover, a similar scheme using the SCA's I2C interface is used for configuration of the TDS chip, which is used for timing and trigger.

Calibration

The calibration procedure consists of various timing and charge calibrations of the front-end electronics. Gain calibration is done by varying the signal input using the internal pulser of the chip. A specific configuration file needs to be loaded on the VMM chips, which is done with the SCA. Calibration of a time-to-amplitude converter is done by skewing the input clock, which is performed by re-configuring the specific on-board with different settings. Baseline and noise level is defined by reading out each channel output with the ADC when no collisions are occurring.

Monitoring

Due to its complexity and long-term operation, the ATLAS detector requires the development of an advanced DCS for the electronics monitoring using the SCA chip, which is installed on the 8000 front-end boards of the NSW. The use of such a system is necessary for the safe operation of the detector as well as to act as a homogeneous interface to all the sub-detectors and the technical infrastructure of the experiment. This system gives us the ability to monitor more than 100000 parameters which include all the power/temperature sensors, on-chip temperature and information, which are connected to the SCA on all the front-tend boards of the NSW. In order to achieve the electronics monitoring via the FELIX-SCA and the SCA OPC UA server chain some steps must be performed in advance. The first step is the creation of the SCA OPC UA Server XML file, which the various sensors connected to the SCA ADC channels are specified. Then, the second step is the initialisation of the FELIXcore and the SCA OPC UA Server, services which are running into the FELIX machine. The third step is the subscription to the SCA OPC UA Server via a WinCC-OA OPC UA Client. Then, another step is the creation of the SCA OPC UA Server XML items into DPEs inside the WinCC-OA. And finally, the monitor of the board parameters is accessible via the SCA electronics control station . The electronics control station has been developed, following the existing look, feel and command architecture of the other Muon subsystems, in order to facilitate the shifter/expert operations. It is mapped onto a hierarchy of Finite State Machine (FSM) elements using the toolkit. For each individual layer, a main panel has been developed, providing the user with useful information, reflecting the state and status of the detector, its vitals displayed in trendplots; while a secondary panel provides supplementary details. A general view of the graphical user interface of the top FSM nodes and their constituents is shown in the Fig. 11 and Fig. 12.

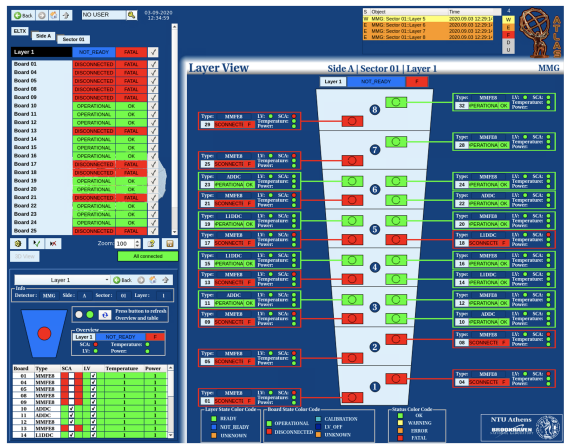


Figure 11: The interface of the Electronics control station and in particular the layer view of the sector which user can navigate and see in a glance the electronics status.

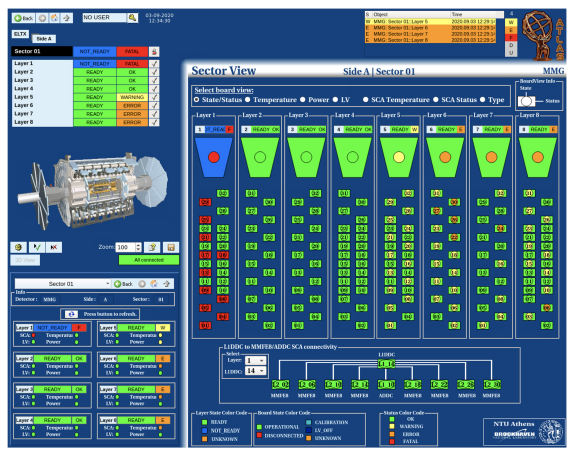


Figure 12: The sector view of the Electronics control station.

Each board's monitoring and conditions can be displayed on the panel. The state and status of the board depends on the status data point element of each board's temperature and power sensors and the communication validity of the SCA OPC UA server. The projects were validated through daily usage from shifters in the commissioning site at a first stage and then through the NSW control in ATLAS Control room.

DCS-DAQ Interaction

As already mentioned above, DCS, calibration and configuration share the common SCA OPC UA Server path for the SCA. Although, the front-end boards which are equipped with the VMM, are using the same SCA ADC input channel

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

for both monitoring and calibration purposes. During the Physics run, the VMM monitor output is equivalent to the VMM temperature but during the Calibration run, the VMM monitor output will display the baseline measurement which is used for the board noice calibration measurements. Thus, during calibration run, VMM shows fake temperature values so a interface plugin between DCS-DAQ should be implemented in order to prevent the DCS to trigger several alarms and warnings. The solution found via the common SCA OPC UA Server and the FreeVariable status which is mainly a user defined OPC-UA item which can be controlled and monitored both by DCS and DAQ back-ends applications. Thus, during the Calibration run, the DCS will monitor the configuration status for this specific VMM of the FEB and the alarm will be disabled corresponding FEB's VMM, as shown in Fig. 13. In addition, more DCS-DAQ interaction tools are on-going in order to monitor the FELIX status, the GBTx alignment information and various FELIX infrastructure parameters.

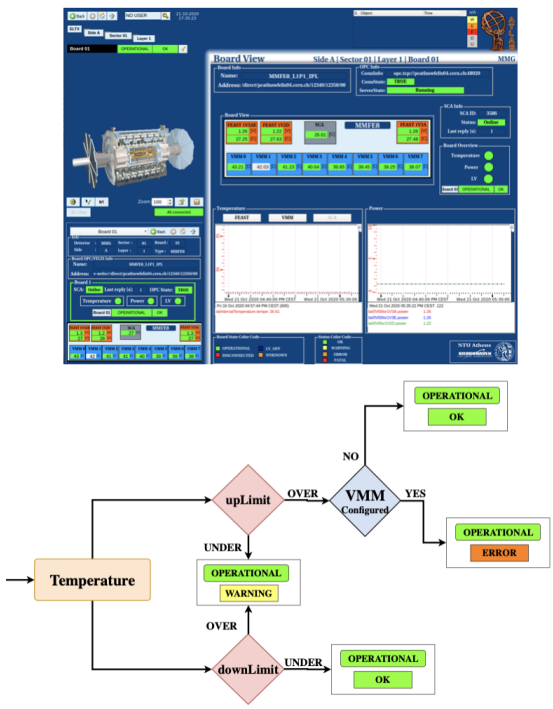


Figure 13: Top: Overview of the board panel of the electronics control station which shows the VMM temperature alarm disabled using the configuration status of the FEB's VMM via the SCA OPC UA Server. Bottom: A flow chart of the logic for the status and state definition based on the VMM temperature and the configuration status.

ACKNOWLEDGEMENTS

This work was funded in part by the U. S. Department of Energy, Office of Science, High Energy Physics under Contracts DE-SC0012704, DE-SC0009920.

We acknowledge support of this work by the project "DeTAnet: Detector Development and Technologies for High Energy Physics" (MIS 5029538) which is implemented under the action "Reinforcement of the Research and Innovation Infrastructure" funded by the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (NSRF 2014–2020) and co-financed by Greece and the European Union (European Regional Development Fund).

REFERENCES

[1] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", *J. Instrum.*, vol. 3, p. S08003, August 2008. doi:10.1088/1748-0221/3/08/S08003

[2] ATLAS Collaboration, "New Small Wheel Technical Design Report", CERN-LHCC-2013-006, ATLAS-TDR-020, https://cds.cern.ch/record/1552862

[3] T. Alexopoulos *et al.*, "Performance studies of resistive-strip bulk micromegas detectors in view of the ATLAS New Small Wheel upgrade", *Nucl. Instrum. Meth. A*, vol. 937, pp. 125-140, 2019. doi:10.1016/j.nima.2019.04.050

[4] P. Tzanis, "Electronics performance of the ATLAS New Small Wheel Micromegas wedges at CERN", *J. Instrum.*, vol. 15, p. C07002, 2020. doi:10.1088/1748-0221/15/07/C07002

[5] G. Iakovidis, V. Polychronakos, and G. de Geronimo, *VMM — An ASIC for Micropattern Detectors*, in *Proc. MPGD2015*, Trieste, Italy, Oct 2015, pp.07001. doi:10.1051/epjconf/201817407001

[6] W. Wu, "FELIX: the New Detector Interface for the ATLAS Experiment", *IEEE Trans. Nucl. Sci.*, vol. 66, no. 7, pp. 986-992, July 2019. doi:10.1109/TNS.2019.2913617

[7] P. Moreira, A. Marchioro, and K. Kloukinas, "The GBT: A proposed architecture for multi-Gb/s data transmission in high energy physics", in *Proc. of the Topical Workshop on Electronics for Particle Physics*, pp. 332-336, 2007. doi:10.5170/CERN-2007-007.332

[8] A. Barriuso Poy *et al.*, "The detector control system of the ATLAS experiment", *J. Instrum.*, vol. 3, p. P05006, 2008. doi:10.1088/1748-0221/3/05/p05006

[9] A. Caratell *et al.*, "The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments", *J. Instrum.*, vol. 10, p. C03034, 2015. doi:10.1088/1748-0221/10/03/C03034

[10] O. Holme *et al.*, "The JCOP framework", in *Proc. ICALEPCS'05*, Geneva, Switzerland, paper WE2.1-6O, Oct. 2005. https://jacow.org/ica05/proceedings/pdf/O3_005.pdf

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

MACE CAMERA ELECTRONICS: CONTROL, MONITORING & SAFETY MECHANISMS

Saurabh Kumar Neema, Shikha Srivastava, Hariharan J., Sandhya Mohanan,
Saju Joy, Padmini S., Anita Behere
Bhabha Atomic Research Centre, Mumbai, India

Abstract

MACE Telescope installed in Ladakh Region of India comprises of many functionally diverse subsystems, Camera being the most important one. Mounted at the focal plane of 21 m diameter parabolic reflector dish, event driven Camera system comprises of 1088 PMTs, with 16 PMTs constituting one Camera Integrated Module (CIM). Central Camera Controller (CCC), located in Camera housing, manages and coordinates all the actions of these 68 Modules and other camera subsystems as per the command sequence received from Operator Console. In addition to control and monitoring of subsystems, various mechanisms have been implemented in hardware as well as embedded firmware of CCC and CIM to provide safety of PMTs against exposure to ambient bright light, bright star masking and detection and recovery from loss of event synchronization at runtime. An adequate command response protocol with fault tolerant behaviour has also been designed to meet performance requirements. The paper presents the overall architecture and flow of camera control mechanisms with a focus on software and hardware challenges involved. Various experimental performance parameters and results will be presented.

INTRODUCTION

The MACE Telescope is a 21m diameter gamma ray telescope installed at Hanle in Ladakh, India at an altitude of 4270m above sea level, the highest for any Imaging Atmospheric Cherenkov Telescopes (IACT) based telescope. Primary objective of such IACT's is to detect High energy cosmic gamma rays emanating from various galactic and extragalactic sources. MACE Telescope detects very faint and narrow (5-10 nS) Cherenkov light generated by Extensive Air Shower when High Energy Gamma rays (20 GeV – 10 TeV) interact in the earth's atmosphere. The Imaging Camera is an essential subsystem of the Huge Telescope system and has been designed with state-of-the-art technologies for High Speed Data acquisition within the constraints of space, weight and power in such a way that the entire electronics for analog signal processing, digitization, triggering and event building is fully integrated into the camera body. Only Power supply and Network cables are connected between Ground station and Camera. Control and monitoring aspects of highly compact camera electronics is discussed in the current paper. Figure 1 presents the latest photograph of MACE Telescope during observation.

Figure 2 describes the block diagram of various subsystems of Camera Electronics. Second level trigger Generator (SLTG) detects time-space coincidence across nearby

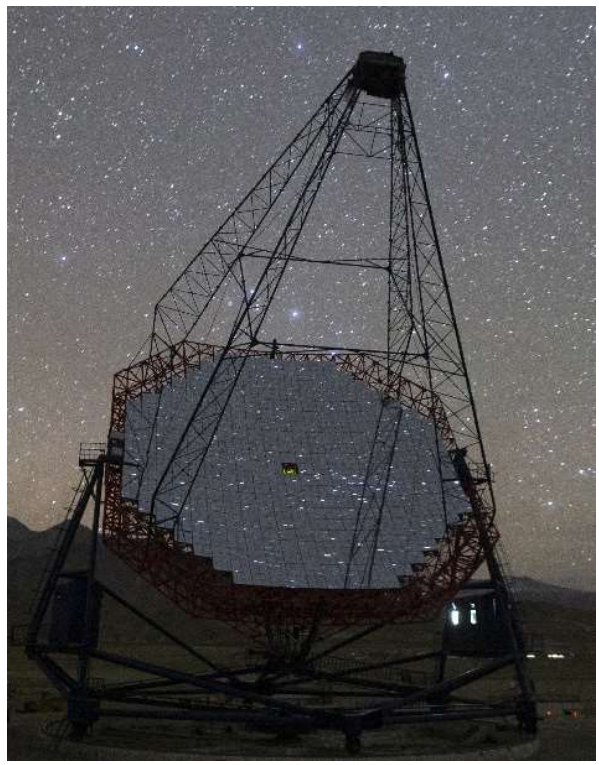


Figure 1: MACE Telescope during observation.

pixels in trigger regions based on the first level Trigger information from various CIMs and generates system-wide Trigger signal. After receiving the Trigger signal, CIMs start acquisition and send event data to the Data Concentrator (DC). Efficient data processing algorithms [1] have been implemented in CIM and accumulated charge data for each pixel along with profile of hit pixels is formed in the CIM data packets. DC collects these data packets from all the CIMs and prepares camera event packets. Event data from camera is sent via 1 Gbps Ethernet link to the Ground station. The co-ordination, control and monitoring of various subsystems in a reliably deterministic manner is a challenging task.

Other subsystems which are part of the camera electronics are Lid Controller, Temperature monitoring system and LED/Sky Calibration system.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

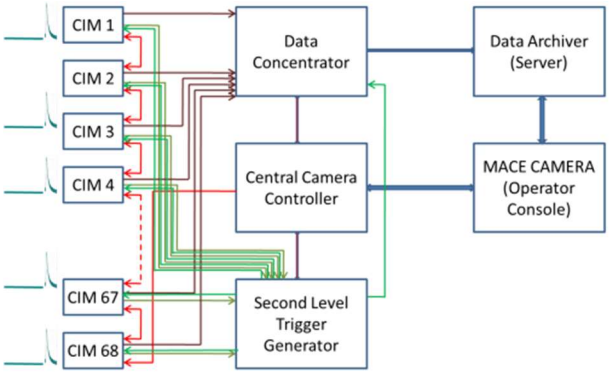


Figure 2: MACE Camera block diagram.

MACE CAMERA CONTROL SYSTEM ARCHITECTURE

Requirements of the System

The main requirements of MACE Camera Control system is to set system configuration, monitor operating parameters, coordinate all functions through command/ response protocol and take corrective / recovery action to ensure safe and stable operating conditions during a sky observation.

The settable configuration parameters include pixel related settings like Bias voltage, Discrimination Threshold etc. within CIM. For STLG the configuration includes Trigger Configuration, Trigger mode etc. and for DC it includes Acquisition mode, Active data modules etc. The operating parameters such as Anode Current (AC), Single Channel Rate (SCR) etc. that indicate light seen by the PMTs are monitored at higher rate of 5sec to ensure corrective action to limit exposure of PMTs to excessive light. Set parameters such as PMT Bias Voltage (HV), Temperature, Power Supply Status etc. are monitored at 1min to ensure stable operating conditions.

Other operation involves Lid controller to safe and reliable shutter open/close based on the command from ground station and Temperature monitoring to monitor temperature at different locations inside the camera.

Hardware Implementation of the System

Each Camera Integrated Module (CIM) contains programmable HV generation module, Pre-Amplifiers, Amplifiers, Pulse discriminators, Scalar counters for individual channels [2]. Each module also houses First Level Trigger generators, scalar counters and Digitization circuits. Extensive Data acquisition, Control and monitoring analysis has been carried out on CIM modules [3]. All these 68 Modules and other subsystems are controlled by Central Camera Controller (CCC) [4]. CCC receives control commands from ground station over Ethernet and accordingly controls various subsystems of Camera Electronics and transmits regular Health monitoring information to the ground station.

Camera Shutter open/close mechanism is also implemented in CCC embedded software as well as FPGA firmware. Two shutters (2m x 1m each) are moved by DC Brushless motors, which are controlled by CCC FPGA firmware in Voltage control mode. It regularly checks for various limit switches and varies the speed of motor to optimize the time to open/close. Various failsafe mechanisms are implemented in firmware to avoid deadlock condition. There is also a mode to operate shutters manually during maintenance schedule. Figure 3 shows the rear view of MACE Camera Electronics.

There are two control links between CCC and CIM modules, which are connected in multi-drop topology. First control link, based on RS485 standard, is used to sense the status of Input power supplies to CIM and accordingly

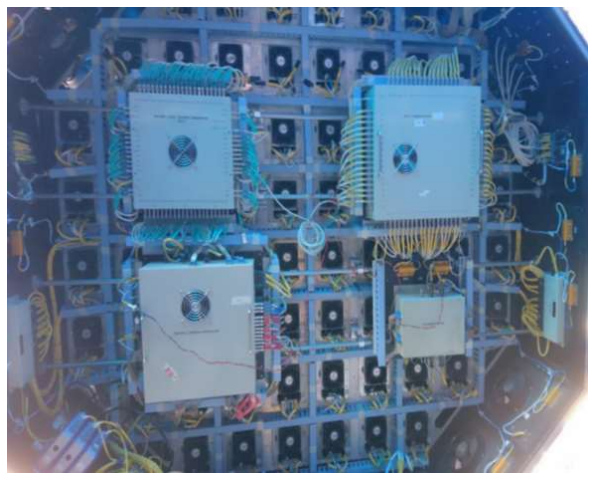


Figure 3: Rear view of MACE Camera Electronics.

Switch ON/OFF of CIMs can be performed. CIMs can be reset using this link. This link is used for sequential powering of CIMs, which reduces sudden power surge associated with powering of all CIMs together. Second control link is based on I2C protocol and it is used to control overall functioning of CIM module including individual channel HV values, Discrimination threshold for each channel, First level Trigger logic modes, Digitization modes etc. This link is also used to collect and send health parameters of the CIM including HV read back value, Anode current, Discrimination Threshold read back, Scalars of various input channels, readings of 5 temperature sensors inside CIM etc. I2C bus operates at 100 kbps speed. Local microcontroller firmware controls various functions of the CIM. Detailed description of the firmware is given in the next section. The control link between CCC and Data Concentrator is RS-232, operating at 115200-baud rate; through this link, DC is set into proper mode of operation and various status flags are read at regular interval. Control link between CCC and SLTG is based on SPI Interface operating at 1 MHz clock; through which SLTG is set into particular operation mode and various status flags are read at regular interval. Figure 4 describes the block diagram of various control links.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

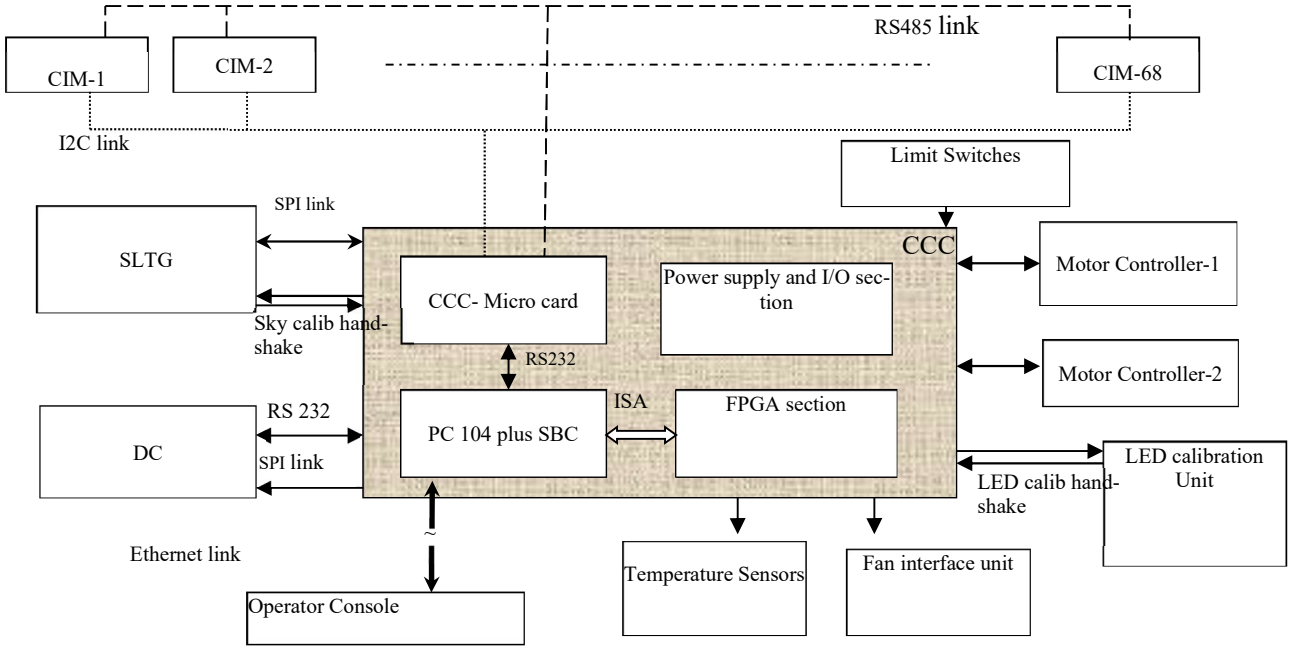


Figure 4: Control and monitoring system of MACE Camera.

Software Implementation

For control and monitoring of various subsystems and to provide overall safety of MACE Camera, embedded firmware of CCC and CIM plays an important role.

Each CIM has a controller card, referred to as LTC Micro board, which controls various functions of CIMs. It monitors telemetry parameters of the CIM like Anode Current, SCR, HV, DT, board temperature etc. Additional functions of control of PMT HV bias and LVDS link for event data transmission to data concentrator and sequential powering up of camera electronics are also performed. The LTC Micro module communicates with CCC on a multi-drop, differential I2C network with CCC acting as master and CIM as slave. System and experimental configuration settings are received upon initialization of the CIM. Post configuration, different telemetry parameters are acquired from the CIM at regular intervals on request- response basis.

The Microcontroller firmware is designed to be multi-tasking in order to effectively monitor the critical parameters of CIM and to service asynchronous telemetry data requests and other commands from the CCC. Periodically, CIM parameters of HV, DT, AC and SCR are acquired from the respective hardware modules. A correction algorithm is run on the acquired data, which analyses the current data with set limits for each parameter from the pre-loaded configuration. The algorithm iteratively evaluates and sets each pixel's state based on the current value of monitored parameter. Based on the algorithm, a pixel identified to be persistently in an abnormal state for a set duration is permanently disabled to safeguard the PMT. Concurrently, a dedicated task responds to commands from CCC.

ISR for controlling I2C operations has been designed and exhaustively tested to handle all conditions of the bus state [5]. However, in situations where the CIM becomes unresponsive over the I2C network, a RESET command from CCC can be sent over an auxiliary link to reset the microcontroller to its initial state.

Central Camera Controller (CCC) is the central entity that manages and coordinates all the actions of the camera. The CCC interfaces with the subsystems for control, configuration, monitoring, error handling, and recovery to ensure that fault tolerant and reliable observations are obtained. CCC software has a layered architecture consisting of Managers dedicated for specific functionality at each level, which interact among themselves through event notifications. The top layer consists of important modules like Command Parser, Telemetry Manager, Sky Calibration Manager and Sync Loss Manager for carrying out dedicated functionalities like managing actions on Camera subsystems by parsing the command received from MACE OC, and serializing the final response to MACE OC after execution; periodic health monitoring of CIMs, collection of diagnostics information from all the subsystems and sending them to OC; facilitating the sky calibration cycles during data acquisition; and error handling and recovery of Camera during event data acquisition from synchronization errors. Managers at top layer uses the lower level Communication Managers for interaction with different subsystems involved.

Apart from basic functionalities of control and monitoring, CCC software also provides fault tolerance, error handling and recovery of Camera at runtime [6]. To ensure high availability of the entire telescope, CCC allows failures of certain number of CIMs, within acceptable limits that is user configurable. On error response from CIMs, on

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

timeout or incorrect response, after 3 retrials of the command, it disables and power off the faulty module. The scheme allows the telescope to continue to operate with useful data collection making the system fault tolerant in nature. It provides error handling mechanism by categorizing different types of errors for each of the sub-actions involved for Commands from OC including timeout and communication error. Some of the errors are handled and recovered by retrials of command, while others are reported at OC.

An adequate command response protocol between CCC and CIM has been established to meet the stringent performance requirement of acquiring periodic telemetry requests from 68 CIMs every 5 seconds. CCC sends some of the commands for CIMs over I2C link in a Broadcast mode for parallel execution of the commands, wherein one of the active CIMs, set as an acknowledgement module, responds with the result of the command. This broadcast scheme highly increased the responsiveness of the system by reducing the effective system response time for overall observation run.

CONTROL AND MONITORING FEATURES

Bright Star Masking

In the field of view of certain Gamma ray sources, various bright stars are also visible. These bright stars emit light, which create very high Single Channel Rates (SCR) in few of the nearby pixels, resulting in generation of spurious triggers from the Trigger Generator and compromising the recording of actual gamma ray events. In order to avoid such spurious triggers, a Bright star masking Algorithm has been implemented in the firmware of CIM. This algorithm monitors SCRs of each pixel every second, and if the value crosses a set Threshold limit, the particular pixel is disabled from participating in generation of First Level Triggers. This way the pixel remains active and collects charge information during actual gamma ray event caused by other pixels without generating spurious triggers. Hysteresis of 70 % has been implemented in the Algorithm itself, due to this the pixel starts participating in trigger generation when the SCR values reduces to 70% of set Threshold value. The following graph shows the pattern of Bright start masking in the case of pixel no. 855 during the observation of source MrK501 on 29/08/2021. For the day's experiment, Threshold for Bright star masking was kept as 1 MHz. During the observation period, the algorithm got activated for few of the pixels. For the sake of simplicity pattern of only one pixel is presented in Fig. 5.

PMT Light Exposure Protection

Photo multiplier Tubes are an indispensable part of Camera electronics and it is of paramount importance to preserve them from overexposure to unwanted light, which may reduce their productive life. During the source observation, various vehicular movements or bright stars cause drastic increase in Anode currents of individual PMT based pixels. In order to protect PMTs from exposure to unwanted light sources, an algorithm has been implemented in the firmware of CIM which continuously monitors Anode current (AC) and SCRs of individual pixel every second. If any pixel crosses the set value of AC for a particular number of times called UCT, the algorithm temporarily disables the particular pixel by reducing the bias voltage to a safe value. The algorithm re-enables the pixel after a set time interval called Reactivation Time (RT) and checks for AC again for UCT and accordingly takes a decision to keep it enabled or temporarily disabled. Few such attempts are carried out which are set by Maximum Retrial Count (MRC), after that the pixel is permanently disabled for the day's observation. Graph in Fig. 6 describes the behaviour of the pixel protection algorithm. Here, the y-axis describes the PMT Anode current and x-axis describes the Telemetry sample number. The threshold for Anode current is set at -5 μA. Value of UCT and RT is set as 4 and 30 seconds respectively.

Event Data Synchronization Loss Detection and Recovery

After receipt of Master Trigger signal from SLTG, all the CIMs acquire event data independently and transfer the event data in a predefined packet format through unidirectional point to point LVDS links to DC. DC Firmware then reads the event data from individual channel buffers and builds the Event data packet. It is of paramount importance that DC event packet shall have all the event packets from CIM, which are due to the same cosmic event and the corresponding event numbers across all CIMs are synchronized. To ensure that there is no event misalignment due to any data corruption during transmission or processing, Event synch-loss detection and recovery mechanism has been implemented in DC and CCC. Data Concentrator checks for event marker at every DC event in order to detect any data processing error. Additionally, DC regularly checks for CIM Event packet nos. for all the CIM event packets in a DC event packet. If there is any mismatch in the CIM event nos., then DC informs Loss of Data Synchronisation to CCC. On receipt of Synch loss info from DC, CCC stops the ongoing data acquisition and restarts a fresh data acquisition after resetting all event counters. This automatic recovery of Synch loss takes approx. 5 seconds and it is a very rare phenomenon in actual observations. Even if it occurs, it gives advance indication to some hardware related issue like data cable malfunction or other hardware issues that can be sorted out with pro-active fault diagnosis.

Regular System Calibration

During the course of the observation, there is huge variation in night sky background light due to various factors. For better resolution of acquired charge on the PMT during source observation, effect of this background light on pixel accumulated charge has to be considered. Regular system calibration runs are carried out during the source observation every 5-15 min interval so as to assess the extent of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

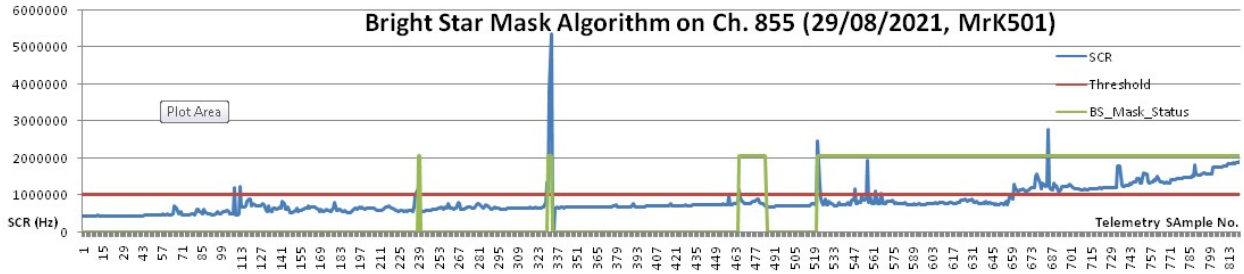


Figure 5: Bright Star Mask algorithm trend on pixel number 855.

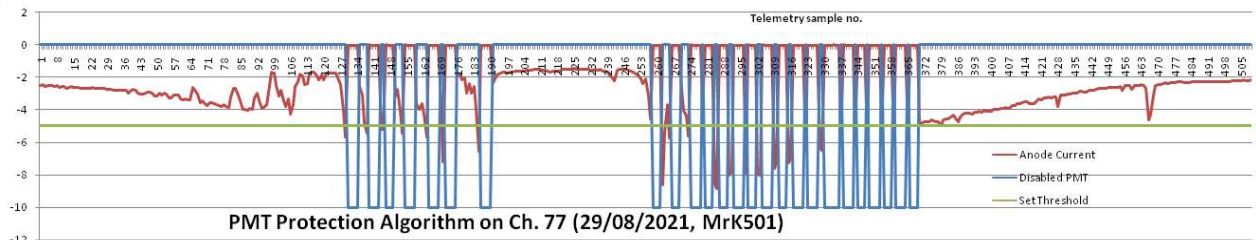


Figure 6: PMT Protection algorithm pixel number 77.

Commands Result | Alarms | POST Status | Version Information | LID Status

LMID	HMID	LTCMicro	LTCFPGA	PS	SPI	HVI2C	SDDLeft	SDDRight	FLT	Temperature	HV_Rb	AC_Rb	Thresh	Scalar	Calib	VersionInformation
60	70	0 0	0 0 0 0	b4	c	0	0 0 0 0	0 0 0 0	0	2b 2b 2a 2a 15	0 0	0 0	0 0	ff ff 0	0	71 21 21 21 21 ff ...
61	65	0 0	0 0 0 0	b4	c	0	0 0 0 0	0 0 0 0	0	2b 2b 2b 2a 15	0 0	0 0	0 0	ff ff 0	0	71 21 21 21 21 ff ...
62	20	0 0	0 0 0 0	b4	c	0	0 0 0 0	0 0 0 0	0	2b 2a 2a 2a 18	0 0	0 0	0 0	ff ff 0	0	71 21 21 21 21 ff ...
64	45	0 0	0 0 0 0	b4	c	0	0 0 0 0	0 0 0 0	0	2b ff 2a 29 18	0 0	0 0	0 0	ff ff 0	0	71 ff 21 21 21 ff 20
65	9	0 0	0 0 0 0	b4	c	0	0 0 0 0	0 0 0 0	0	2a 2b 2a 2b 17	0 0	0 0	0 0	ff ff 0	0	71 21 21 21 21 ff ...

Figure 7: POST Status on operator console of MACE Telescope (08/07/2021).

variation of night sky background light. At the set Interval, CCC gives commands to SLTG, which in turns generates fix number of self-triggers at a set frequency due to which all the pixels record charge collected due to background light and such events are tagged as SKY calibration events in DC Event data packet. Similarly, PMT gain calibration is also carried out at a regular interval just after the SKY Calibration cycle, during this CCC instructs LED driver (mounted at the centre of reflector dish) to generate fix number of LED pulses at a set frequency. All the PMTs collect charge due to LED pulses fired from the centre of reflector dish. Gain of all the PMTs is calibrated with respect to the charge calculated in the Reference PMT.

Safety & Diagnostics Features of the System

Due to harsh weather conditions in Hanle, Temperature in winters can go up to -30 degrees Celsius. CCC is designed to operate in such harsh environments and initially only CCC is powered on. It acquires the temperature from 8 Temperature sensors mounted inside the camera body and 2 mounted on the outer body of Camera. Temperature reading from all these is taken into consideration before powering ON of other subsystems of the camera in winter season. Passive heaters are provided inside the Camera to bring Camera Temperature to an operating range. Only after attaining the operating temperature, various subsystems of camera are switched ON. At the time of powering ON of Camera electronics, each subsystem performs its own Power on Self-Test (POST). After power ON CIM checks for various memory locations, setting and read back of discrimination voltages, ADC baseline read back, firmware version mismatch, Temperatures etc. If there is any error in POST STATUS from CIM, the particular CIM is switched OFF and does not participate in observation. One such scenario is presented in Fig. 7 where one of the board firmware version information of CIM 64 was not recorded correctly at the time of POST so CIM 64 was switched OFF for the particular experiment. The system also performs various link tests like Auxiliary Link Test, I2C link test and Data link test etc. and takes corrective actions before proceeding to the observation schedule. Regular health parameters of CIM are recorded and sent to the ground station. There are 5 Temperature sensors inside each CIM. CCC monitors each of them at regular intervals, if the temperature of any CIM crosses the set value, CCC can take corrective actions.

CONCLUSION

MACE Telescope is operational since early 2021 and regular trial source observations are being carried out. Initially stability of various configuration parameters was verified by long trial runs in close lid conditions and parameters were found to be stable over a long period of time e.g. 4-5 Hrs. various control and monitoring features of Camera

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Electronics have been rigorously tested in real source observation condition. Implemented algorithms like self-diagnostics, safety, PMT protection and Telemetry monitoring etc. are performing as per the design specifications.

ACKNOWLEDGEMENTS

Authors would like to thank the colleagues from Astrophysical Sciences Division, BARC Mumbai for their support in various activities related to MACE Camera Electronics. Authors would also like to thank the staff of MACE Telescope site, Hanle for their support in carrying out the observations in the challenging weather conditions.

REFERENCES

[1] Saurabh Neema, Saju Joy, K. Jha, Anita Behere, P.K. Mukhopadhyay, S. Ritt; "Design of High-Speed Event Acquisition for the MACE Telescope using DRS ASIC", *National Symposium on Nuclear Instrumentation*, pp. 170-173, 2010.

[2] Saurabh Neema, Saju Joy *et al*; *"Sixteen pixel integrated electronics for imaging camera of MACE Telescope"*, Proceedings of National Symposium of Nuclear Instrumentation -2013, NSNI13_236, pp 1-5.

[3] Nilesh Chouhan, Sagar Godambe, Saurabh Neema *et al*, *"Performance evaluation of the Camera Integrated Module for the MACE Telescope"*, Proceedings of National Symposium of Nuclear Instrumentation-2013, NSNI13_236, pp 1- 5.

[4] Saju Joy, Saurabh Neema *et al*, *"Hardware architecture of Central Camera Controller for MACE Telescope"*, Proceedings of National Symposium of Nuclear Instrumentation - 2013, NSNI13_236, pp 1-5.

[5] *UM10204I2C-bus specification and user manual* Rev. 6 - 4 April 2014

[6] S. Srivastava, A. Jain, P.M.Nair, P.Sridharan, *MACE camera controller embedded software: Redesign for robustness and maintainability"*, Astronomy and Computing, Volume 30, January 2020, https://doi.org/10.1016/j.ascom.2019.100358

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

TARANTA, THE NO-CODE WEB DASHBOARD IN PRODUCTION

M. Eguiraun*, V. Hardion, Y. Li, M. Saad, A. Amjad, J. Rosenqvist, L. Nguyen, J. Forsberg
MAX IV Laboratory, Lund, Sweden
M. Canzari, V. Alberti INAF-OAAB, Teramo, Italy
H. Ribeiro, Atlar Innovation, Portugal
V. Alberti, INAF-OATs, Trieste, Italy
A. Dubey,Persistent Systems, Pune, India

Abstract

The remote control and monitoring of accelerators and experimental setup has become essential when remote work has become the norm for the last two years. Unlike the desktop user interfaces which have been developed for the use from physical workstations, web application are naturally accessible remotely via the ubiquitous web browsers. On the other hand, Web technology development requires a specific knowledge which has yet to be disseminated in the control system engineering and desktop frameworks still have the benefit of rapid, and easy development even for the non-specialist. Taranta Suite is a collection of web applications jointly developed by MAX IV Laboratory and the SKA Observatory, for the Tango Control System. In line with the "no-code" trend for the users, truly little knowledge of web technologies is needed. An operator can create a graphical user interface on-the-fly and can share it instantly. Authentication and authorization ensures that the right access level is given to the user. This paper will describe the system, the details of its implementation, and the first usage at the different facilities.

INTRODUCTION

There is no doubt about the usability and the optimised user experience of web interfaces and applications in everyday life situations. The last decade has seen an explosion of these kind of developments. New tools, new frameworks and new wide-spread applications have gained fame and number of users and slowly displaced traditional desktop applications. However, scientific environments are usually built on traditional and well known infrastructure, lagging behind software innovation and trends. MAX IV and SKA facilities were pushing and promoting the usage of web applications in their respective communities and in 2019 they joined efforts and gave birth to Taranta, a web application for building user interfaces in a Tango ecosystem [1]. The tango community renamed it from the previous Webjive name [2].

This new application is profiting from recent years of development in User Experience (UX) and User Interface (UI) web frameworks. It provides an out of the box, modern and stylish environment for accessing the most important functionality of a Tango device, from what is called the Device View [2]. Simple and powerful enough for fast access, however, the most remarkable functionality is the Dashboard View. This element is where the user can create its own user interface by drag and drop components (which are called Taranta Widgets) and easily configure and link them to Tango devices. This is where the idea of No-Code plays a fundamental role, Taranta leverages the UI development to the end user.

NO-CODE PARADIGM

The lead time to get a new user interface is usually very long. The software development and UI design have to follow a number of established stages starting from the user requirement gathering to the usage of a final product. The no-code trend define a way for any end-user to develop their own software and then to make it available immediately to a larger audience of the same end-user group [3]. This way, user of a no-code system doesn't need to be knowledgeable in software development and how the software is deployed to be able to add value into the system. All the infrastructure is completely transparent and does not prevent fulfilling their will.

There is a lot of advantage for the users to bypass the traditional software development chain. First of all, as users themselves, they know exactly what the software should look like and which feature is expected. The users own the requirements like in any development method, although in a no-code system many filters are avoided since the users develop their own product. Writing down the requirements, understanding the specification and code development throughout different persons are some examples of filters which attenuate the original idea.

The concept of accelerating the software development has started years before the no-code trend arrived. Prototype generated by sketch of the Graphical User Interface (GUI) in the Rapid-Application Development (RAD) [4] made a step closer to the final product by developing only the functionalities. An extension of the same concept appears in the early 2000s for the creation of UI based on the definition of the domain i.e in the model development driven (MDD) all the structure of the software (Model) and part of its implementation is generated from the requirement. Visual programming language helps the power-users with enough software skills to design a system rapidly for their local applications i.e workbench like LabView can produce application with no manually written code code. The dissemination to a larger group of users can be delegated later to the software engineers.

* mikel.eguiraun@maxiv.lu.se

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

In our Scientific Facilities and in particular in the Control System, the software development has followed the same trend all the long to the industry standard evolution. Application like Taurus Designer [5] for the Tango Control System and Control System Studio for EPICS [6] allows to leverage the skills of the scientist to create GUI on top of their Control System. No-code introduces another level of independence for the user and rely on them to also publish their software to an entire group of people without involving any IT staff for the deployment, which is a large cost of a software. Not only can the end-user benefit from this possibility but also the they can leverage the use of technologies traditionally separated, e.g. web technologies and control system frameworks.

The No-code platform are designed from the web technology for its low-cost of deployment of application. Only one computer server running a web application is necessary to allow many clients to access the program. On the contrary, a desktop based application has to be deployed on every single workstation which is usually a long and not user-friendly process.

Although No-code is completely oriented to simplify the end-user development, the software developers are still needed to program the elementary bricks of the application.

TARANTA

A Taranta application is composed of two main elements: the backend and the frontend application. The backend provides a GraphQL API to a Tango control system. The client is a web application that provides a generic tango device view, similar in functionality to the well known Jive desktop application, but it also provides a dashboard. A blank canvas for the users to build their User Interfaces.

Figure 1 displays how Taranta components are linked together. It is composed of several applications, or microservices, this way each component is developed and tested independently. The next sections will describe the most relevant aspects of the main elements.

TangoGQL

GraphQL is a modern query protocol for the application layer developed by Facebook [7]. It provides a unified interface between the client and the server for performing data oriented actions e.g fetching and manipulation. It removes the need of having multiple endpoints for data manipulation to/from the server and instead returns the data what the client asks for over a single API endpoint. The advantage of this approach is that the client asks explicitly for what it needs. In addition, due to its schema-based implementation, the extension of the API does not change the API interface so that compatibility between versions is simplified.

TangoGQL [8] is an implementation of GraphQL over Tango. It provides a communication based on web-socket [9] for subscribing to asynchronous attribute value events, and a GraphQL interface to the Tango database. The standard operations over a Tango device are supported, for example

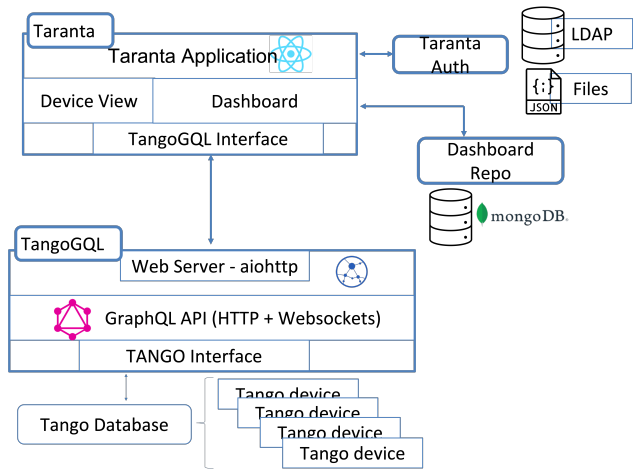


Figure 1: Structure of Taranta, at the bottom and connected to the tango ecosystem there is the TangoGql interface, to which the Taranta web application talks to. Together with the authentication and repository database systems, Taranta suite is complete.

read/write access to device properties and attributes, and command executions. It is developed in Python 3.9.

Front-End

The front-end of Taranta is a React [10] application that is used both to browse, inspect and control Tango devices and to create and run dashboards, each composed of widgets. It accesses the Tango Control System through the TangoGQL API, the communication between Taranta and TangoGQL is managed by an appropriate frontend component. The first developed component was the Device Viewer, Figure 2,

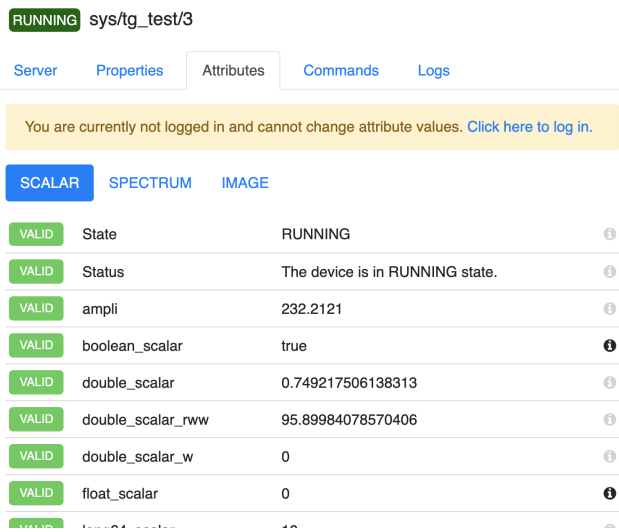


Figure 2: Generic Tango device viewer. Several tabs are available for displaying different types of information, device properties, attributes, commands and logs. The attribute view is further split into the different data types present in the given attribute.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

which provides a generic view for any tango device running in the system. It has a search section, which aims at helping the user navigating the full tango device tree hierarchy. Once the desired device is selected, the right section of the page will display several navigation tabs for accessing device properties, attributes (further split based on the data format), commands and logs. Any user can see everything, but only logged in users can modify attributes or execute commands. The values of the attributes are updated at real time. In addition, the selected device name is appended to the page url, thus, it can be shared with others colleagues to access the same view.

As monitoring tool the Device Viewer is a very fast way to access a Tango device, but it lacks of any kind of customization. In order to provide a configurable user interface the Dashboard View was developed. It belongs to the Taranta front-end but it is a very different user interface. As opposed to the static device view, here the user can create their User Interface starting from a blank page and simply dragging and dropping widgets into it. Each widget comes with its own configuration, depending on the purpose of the widget, but they all provide a Tango device (or attribute) selection.

Taranta widgets A widget is a dashboard component to serve the function of interacting with Tango devices. There are different types of widgets. A command can be sent to tango device through command related widgets. An attribute can be read/write through specific attribute widgets. Using widgets, users are able to monitor attribute changes and take remote control operations.

Available widgets The available widgets fall into different categories according to their functions: labels, attributes, commands and plotting. The label widget is mainly used to display static information. It is basic and also flexible with a hyperlink feature, that can be configured by the user to switch between different dashboards. It enables well-structured dashboards.

The attribute widget family is used to display the value of interested attributes, which also includes the device status. The displayed information and text size can be customized to user settings. Several different widgets are available depending on the data type of the attribute (numeric, boolean, etc.), but also some widgets provides extra functionality for example the SimpleMotor and SardanaMotor widgets, where movement steps can be defined and move commands issued.

A command widget enables users to send commands with parameters to tango devices. It also enables users to customize widget settings based on their needs. Again, several widgets are available for different command arguments and data types.

The array type attributes have its own set of widget. For example, a spectrum widget is a special attribute widget, which allows plotting of 2-dimensional attributes. Multiple spectrum attributes can be plotted on the y-axis against one spectrum attribute on the x-axis . The plots are updated when a new attribute value is pushed in by its associated tango device. The rendering of plot is achieved through Plotly [11], which enables user interaction with the plots including zooming, extracting specific graph parts, and also axes resetting. Several widgets are available providing different visualization options for example heatmaps, scatter plots, table like view, a few others.

The variable selector widget is a new feature which enables the interaction between a widget and a dashboard variable. A dashboard variable allows users to create a parametric dashboard where a running device can be replaced by a variable in widget configuration. Users can associate a device to a dashboard variable and change device in run mode. All widgets having this device which subscribed to this variable will be updated with the new device value.

Currently 25 different widgets are available, and more coming regularly due to requests from our user community. Moreover, development is ongoing to allow users to create new widgets from the existing ones.

Creating a new widget A widget is composed of a definition (also called a widget definition) and a React component. The definition is a declarative object describing the characteristics of the widget and the inputs that it receives.

A widget definition starts with a JSON object. An example of a definition is as follows:

```
const definition = {
    type: "OUR_NEW_WIDGET",
    name: "Our New Widget",
    defaultWidth: "10",
    defaultHeight: "15",
    inputs: {
        device: {
         type: "device",
         publish: "$device",
       },
       position: {
         type: "attribute",
         device: "$device",
         attribute: "Position"
       },
       ....
    }
}
```

In this example a new widget is defined with some styling options as well as the required input fields, in this particular case a tango device must be defined and which attribute of that device should be considered position. In addition, Table 1 describes a brief description about required widget definition keys.

Different types of inputs can be defined through their corresponding input definition interfaces. For example, an attribute can be defined with *AttributeInputDefinition* where the bounded device and attribute name as well as other characteristic keys are set. Different from other inputs, the device name can be published to a variable which is available for other inputs. For TypeScript implementation, the widget inputs can be obtained through *WidgetProps* type mapping, which makes the process much easier to develop a new widget and avoid type inaccuracy [12].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Table 1: Description of Widget Definition Keys

Key	Type	Description
type	string	Type identifier for the widget. Must be unique (e.g. "ATTRIBUTE_PLOT").
name	string	The name of the widget shown to the user (e.g. “Attribute Plot”).
defaultWidth	number	Default width (in number of tiles)
defaultHeight	number	Default height (in number of tiles)
inputs	object	An object where the keys are input names and the values are corresponding to their input definition.

In the corresponding React component, the declared inputs are made available through a prop named *input*. A minimal example of a React component for our new widget would as follows:

```
class OurNewWidget extends Component{
    render() {
        const {
            position
        } = this.props.inputs;

        return (
           <div>
             Position: {position.value}
           </div>
        );
    }
}
export default {
    component: OurNewWidget,
    definition
}
```

This minimal example describes how to create a basic widget. All the rest of the required interaction with the dashboard page as well all the communication with the tango device is already handle by Taranta, no additional development is required. Figure 3 shows the newly created widget in action, in both editing mode for selecting a tango device as well as in running mode displaying an attribute value.

Taranta Dashboard Repository

The dashboard repository is an API for fetching and storing the dashboards. It is crafted from ExpressJS and communicates with MongoDB to store and fetch dashboards. Few endpoints are open to provide data to anonymous users, for example, fetching the dashboards. Whereas other endpoints are restricted and require a valid JWT (JSON Web Token) to perform the respective function.

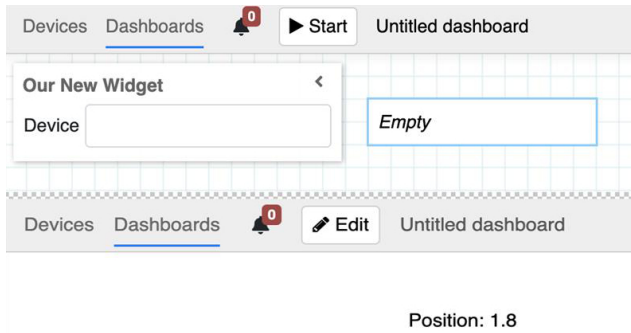


Figure 3: A minimal widget example. The top view displays the widget in editing mode where the users selects the tango device. The bottom view shows the widget in action.

Taranta Auth

Security of web applications has been recently gaining enormous popularity. This could be due to its nature of being accessible from various end-points and therefore of potentially being an entry point to malicious attackers. Furthermore, it could also contain sensitive data and years of progress and their loss can be catastrophic. Several industry standard protocols could be followed at application-level as well as organization-level to enhance the security of web applications and to ensure the data integrity. The Taranta Auth is a micro-service based on NodeJS and provides an authentication service to be use by the Taranta suite for authenticating and authorising users and identifying their shared groups. It uses JSON Web Tokens [13], an open standard to digitally sign information for being transmitted between services. Taranta Auth access an LDAP repository or a JSON file to manage users or to retrieve user information. An anonymous user can run dashboards and can browse and inspect devices. However, to be able to send commands to devices, to change their attributes and to create or modify dashboards, an authentication is required.

USAGE

All facilities involved in the development have a very different experience using Taranta. This is partly due to the stage in which each facility is. MAX IV is in a stable user operation phase while SKA is in building stage. Despite these differences on the particular situation on each facility the development has benefit for both experiences.

MAXIV

Almost every system in MAX IV uses Taranta in one way or another, from accelerator operators to the beamlines, including the whole range of support groups. The operators use dozens of dashboards to monitor the insertion devices, RF and water cooling systems in the LINAC, and also for the status of the personal safety system. These dashboards are primarily focused on monitoring usage, but they also provide control functionalities, e.g. set-point adjustment of the motor movements.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

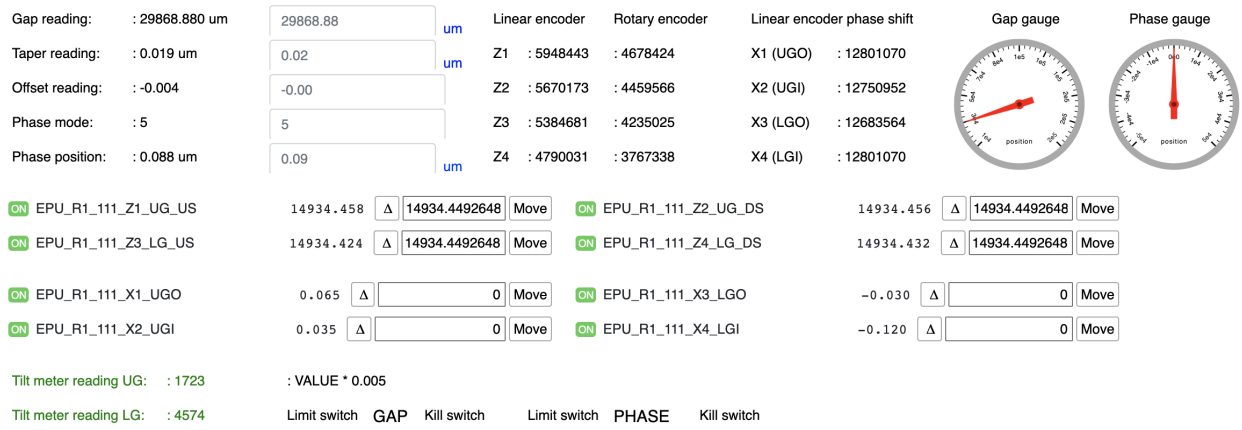


Figure 4: Example of a dashboard for monitorization and control of MAX IV insertion device for a beamline.

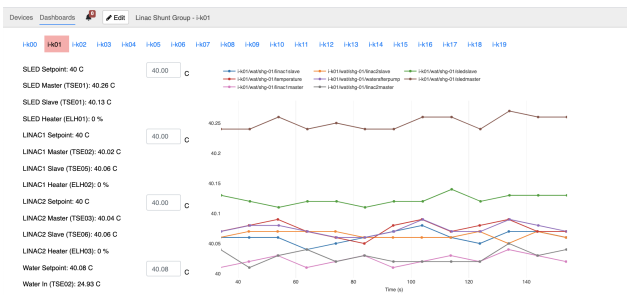


Figure 5: Example of a dashboard for monitorization of cooling systems in the Linac. There is one like this for every klystron.

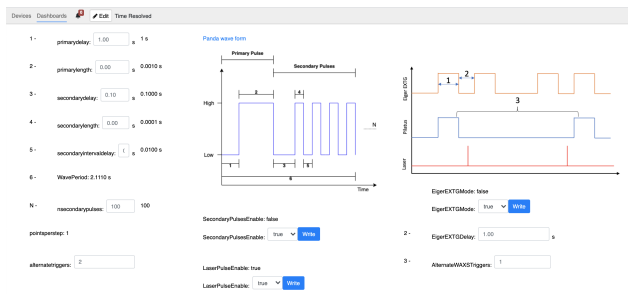


Figure 6: Dashboard for configuration of the timing strategy for the data acquisition at Cosaxs beamline.

Figures 4 and 5 show working examples of production dashboards. The usage approach is different, the first example is more focused on the moveable elements of an insertion device and thus, the user wanted to have the possibility of driving those axis. The second example is oriented towards cooling water trends over time, link widgets above the plot allows to switch to different dashboards.

On the beamline's side, the usage is focused on one hand to provide a simplified overview of equipment by adding into a dashboard only the most critical tango attributes of an equipment. On the other hand, it is also used to provide users with an easy to use interface to configure parts of the experiment. Figure 6 displays a dashboard to configure the timing pulse diagram for the data acquisition in Cosaxs beamline.

There was a slow start with the adoption of Taranta, but thanks to a few early adopters in the machine operators group that provided valuable feedback as well as bug reporting, Taranta is gaining popularity quickly. In addition, the knowledge of widgets development has been improved so we are able to provide new widgets faster than before, which helps gaining trust from our user community.

SKA

Taranta is seen as a tool with a lot of potential within SKA. At the current stage of the project's development, there are different categories of target users for Taranta. First, there are control system developers who create the dashboards and use them to inspect, debug and verify the devices they are developing; secondly, other teams who want to verify the behaviour of a component and use dashboards provided by the team who developed them; and lastly, managers or engineers who may want to check the current status of the system or are tasked with testing the Minimum Viable Product (MVP). Often, Taranta dashboards are used for showcasing the progress in the development of the control system during demos. Depending on their envisaged use, dashboards are expected to have different lifetimes and to require different levels of analysis before realising them.

Figures 7 and 8 show two examples of dashboards realised by SKA teams. The dashboard in 7 has been created by the team that is developing Taranta after some interactions with developers belonging to other teams. It allows to switch ON/OFF the Central Signal Processor (CSP) subsystems as well as to control them by configuring the set of resources to be used and for sending a set of commands for executing a scan (atomic part of an observation).. It also gives fast feedback on the state of subsystems. 8 shows a dashboard that is mainly used for checking the "state readiness" of the MVP for executing certain tasks (and allowing recovery of inconsistent states where possible) whilst conducting interactive end to end test development.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR01

The close interaction with users of Taranta often promotes the development of new widgets or functionalities, such as the timeline widget, or triggers some usability improvements for the tool. Moreover using this dashboard helped developers in other teams to early detect needed changes in the control software and identify bugs.

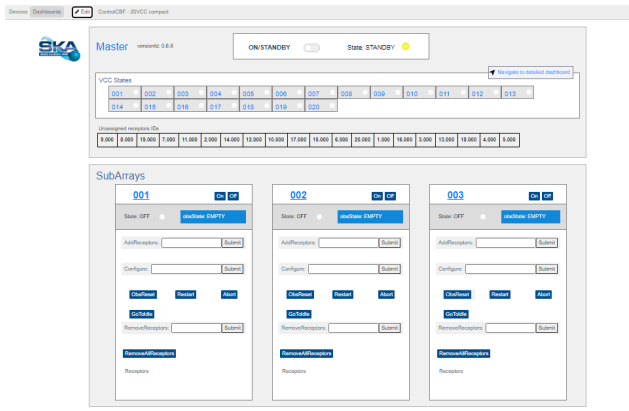


Figure 7: Example of a dashboard for monitoring and control SKA CSP.

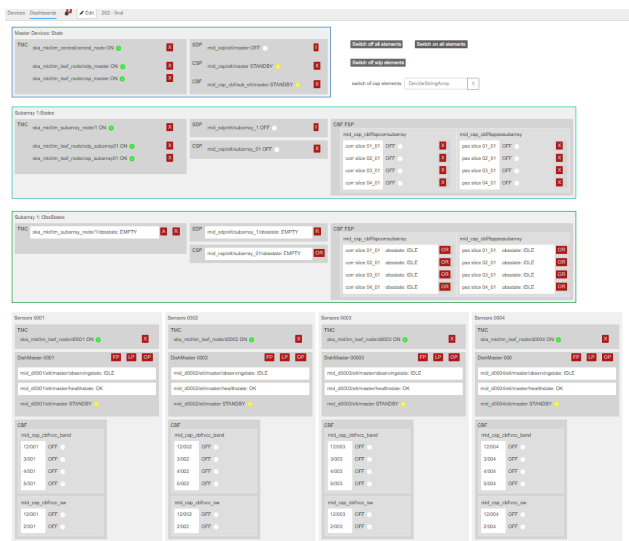


Figure 8: Example of a dashboard to monitor the status of various resources that are part of the SKA MVP.

FUTURE WORK

By the increase of user feedback a lot of new improvement can be done in Taranta. The increased number of available widgets requires a redesigned widget library element, possibly with searching and sorting by type functionality. This would make the life of our users a little bit easier. In addition, several new widgets are under development. For example, a grouping widget, onto which one can drop different widgets and then group it into one entity. This would make designing the interface layout smoother as well as give the users the possibility to create their own widgets. Another interesting widget under development is the synoptic, which aims at displaying svg images and linking svg elements to tango devices [14]. Moreover, optimisation on how and when the application sends updated values to the client, as well as data formatting improvements, specially for images, are on the list. Last but not least important, there will be new widgets coming to complement running the experiments, i.e. moving into web current functionality nowadays scattered over desktop applications and command line applications.

REFERENCES

[1] Taranta Suite home page, https://gitlab.com/tango-controls/web

[2] M. Eguiraun *et al.*, "Web Interface to Tango Control Systems at MAX IV", presented at 12th NOBUGS Conference, BNL, New York, 2018, unpublished.

[3] M. Woo, "The Rise of No/Low Code Software Development—No Experience Needed?", *Engineering*, vol. 6, no. 9, pp. 960-961, 2020. doi:10.1016/j.eng.2020.07.007

[4] J. Martin, *Rapid Application Development*, Indianapolis, IN, USA: Macmillan Publishing Co., Inc, 1991.

[5] Tarus designer, https://taurus-scada.org/devel/designer_tutorial.html

[6] Control System Studio, https://controlsystemstudio.org/

[7] GraphQL, A query language for your API, https://graphql.org/

[8] TangoGQL repository, https://gitlab.com/tango-controls/web/tangogql

[9] The Websocket Protocol, https://datatracker.ietf.org/doc/html/rfc6455

[10] React, A JavaScript library for building user interfaces, https://reactjs.org/

[11] Plotly JavaScript Open Source Graphing Library, https://plotly.com/javascript/

[12] MAXIV Taranta home page, How to create widgets, https://webjive.readthedocs.io/en/latest/writing_a_widget.html

[13] JSON Web Tokens, https://jwt.io/

[14] J. Forsberg *et al.*, "A Graphical Tool for Viewing and Interacting with a Control System", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 681–684. doi:10.18429/JACoW-ICALEPCS2015-WEM309

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

canone3: A NEW SERVICE AND DEVELOPMENT FRAMEWORK FOR THE WEB AND PLATFORM INDEPENDENT APPLICATIONS*

G. Strangolino, L. Zambon, Elettra, Trieste, Italy

Abstract

On the wake of former web interfaces developed at ELETTRA [1] as well as in other institutes, the service and development framework for the web and platform independent applications named PUMA (Platform for Universal Mobile application) has been substantially enhanced and rewritten, with the additional objectives of high availability, scalability, load balancing, responsiveness and customization. Thorough analysis of Websocket limits led to an SSE (Server-Sent Events) based server technology relying on channels (Nchan over NGINX) to deliver the events to the clients. The development of the latter is supported by JQuery, Bootstrap, D3js, SVG (Scalable Vector Graphics) and QT and helps build interfaces ranging from mobile to dashboard. Ultimate developments led to successful load balancing and failover actions, owing to the joint cooperation of a dedicated service supervisor and the NGINX upstream module.

DESIGN RATIONALE

The system consists of a cluster of servers, thereafter synonymously named services, and two client side development environments. One is based on web technologies on browsers. The second is a C++ client library to build native Qt applications. The main objectives of the service design are reliability, security, scalability and accessibility. To satisfy them, a set of state of the art technologies and software serve as the groundwork of the system.

RELIABILITY

The design rationale identifies the principles of a reliable service as follows. The system shall work:

- from any place and platform;
- at any time;
- regardless the number of clients
- included when part of the system is unavailable
- included when the network performance is suboptimal or even subject to charges.

The first requirement ruled out the *WebSocket* technology after an accurate analysis of its assets and liabilities.

WebSocket

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C. [2]

WebSockets are widespread and efficient when handling huge amount of messages from both ends, where duplex communication is continuously involved: Massive Multiplayer Online (MMO) and messaging applications.

The list of liabilities is nevertheless long in our situation:

- WebSockets can be potentially blocked by proxies;
- CORS (Cross-Origin Resource Sharing) [3] related concerns;
- no multiplexing over HTTP/2 (implementing it on both ends is complicated);
- no load balancing;
- susceptible to DoS;
- problems already taken care of in HTTP must be solved ad hoc;
- operational overhead in developing, testing and scaling is increased.

Some proxy servers are transparent and work fine with WebSockets; others will prevent them from working correctly, causing the connection to fail. In some cases, additional proxy server configuration is required.

Load balancing is very complicated. When servers are under pressure and new connections need to be created and old ones closed, the actions that must be taken can trigger a massive chain of refreshes and new data requests, additionally overloading the system. It's not possible to move socket connections to a different server to relieve one under high load. They must be closed and reopened. It turns out that WebSockets need to be maintained both on the server and on the client.

Multiplexing is usually handled by front end HTTP proxies that cannot be handled by TCP proxies which are needed for the WebSockets. Connecting to the sockets and flooding servers with data is a possible eventuality.

Concerning the last weakness in the list, we observe that mobile devices would maintain a WebSocket open by keeping the antenna and the connection to the cellular network active. Battery life would be reduced, heating increased, and, where applicable, extra costs for data usage applied.

SSE

Server-Sent Events (SSE) is a server push technology enabling a client to receive automatic updates from a server via an HTTP connection, and describes how servers can initiate data transmission towards clients once an initial connection has been established. They are commonly used to send message updates or continuous data streams to a browser client and designed to enhance native, cross-browser streaming through a JavaScript API called EventSource, through which a client requests a

* inspiration by Alessio Igor Bogani, Elettra, Trieste, Italy

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

particular URL in order to receive an event stream. The Server-Sent Events EventSource API is standardized as part of HTML5 by the W3C [4].

Server-Sent Events technology analysis underlined some disadvantages; most of them do not appertain to our environment:

- SSE is unfit for duplex communications;
- binary data cannot be sent;
- the number of connections on a web browser is limited;
- mono-directional by nature, additional approaches are required for duplex synchronous operations.

The third limitation in the record must be addressed in web browser based applications, while it does not affect clients of different nature.

Nevertheless, in our context the opportunities offered by SSE outweight the weaknesses:

- SSE is based on HTTP, posing no issues with proxies;
- multiplexing is implemented in HTTP/2
- messages bear an *id*: the server is aware if the client misses one;
- data exchange requires a smaller number of connections;
- on failure, the *EventSource* reconnects;
- the connection stream is based on events, it is read only and goes from the server to the client;
- arbitrary events can be sent.

PUMA framework combines SSE with a channel based event streaming (*Nchan* [5]), thus enforcing the connection economy mentioned in the fourth item in the preceding enumeration. The hitherto introduced benefits offered by an SSE technology over WebSockets cover a fundamental aspect of the reliability essential. In particular, the a*ny place, any time and platform* principles are backed up by SSE. In combination with channels, the system gains scalability in reference to an arbitrarily high number of clients using the service at the same time.

Figure 1 is a representation of the client – server architecture based on Nginx, Nchan and the PUMA service.

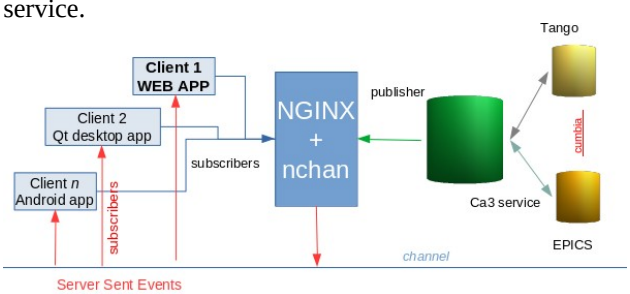


Figure 1: Nginx, Nchan and PUMA service use channels.

The next ingredient in our reliability recipe contributes to maintain the service available to clients even when parts of the system are unavailable. *Nginx* [6] comes into play as a balancer to distribute the load across several instances of the PUMA server and at the same time, through a PUMA *supervisor* service, to manage failover. Moreover, when Nginx and Nchan are combined with *Redis cluster* [7] (Remote Dictionary Server), channels attain high availability and failover capabilities [8].

SECURITY

The goal is to realize a service and its context safe enough as to avoid tunnelling the traffic across a VPN. The latter implies additional software and configuration, hindering usability and reducing battery life, especially on mobile devices. At the moment of writing, the network architecture essential to achieve a proper level of security has still to be laid out. The observations made in the preceding section favour the choice of Nginx and HTTP over WebSockets as far as protection from DDoS (Distributed Denial of Service) is concerned. Nginx can be adjusted to limit the worker processes and connections, as well as the requests rate over time. Additionally, the number of connections that can be opened by a single client IP address can be curbed to protect the downstream PUMA services.

SCALABILITY

A good infrastructure shall be designed with scalability in mind. Reliability and security will be inherently reinforced. Channels to which numerous clients tune in to receive messages are an obvious representative of scalability. Nginx combined with Nchan and Redis offer horizontal scalability to the PUMA service. Figure 2 shows this principle, while Fig. 3 illustrates the initial deployment of the PUMA service architecture at Elettra.

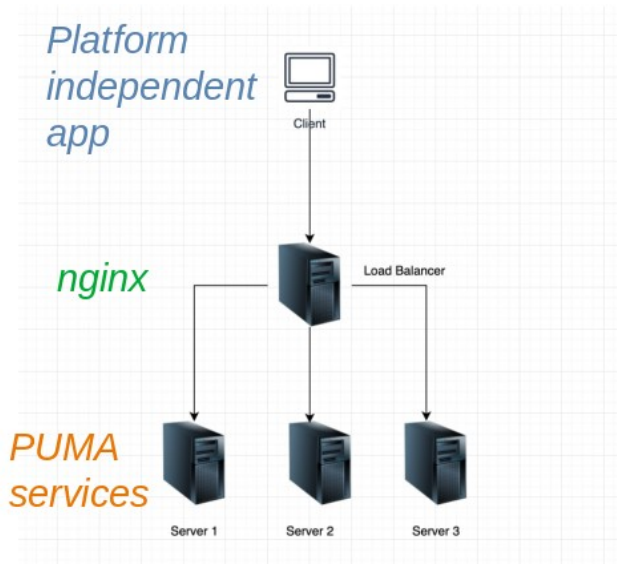


Figure 2: Horizontal scalability with Nginx.

ACCESSIBILITY

The interaction with the service is performed through an open API relying on JSON (JavaScript Object Notation) both on the request and the reply sides. A simplified URL API for requests shall be included in a future release. The API serves the web, the mobile and the desktop applications.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

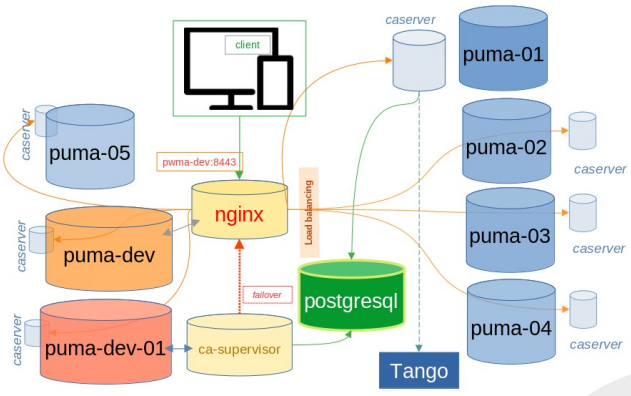


Figure 3: Horizontal scalability as deployed at Elettra.

Reliability, security, scalability and a generic API are the main principles of the PUMA service design rationale. In the following sections, the single components will be described in more detail.

THE SERVICE

The *caserver* is the name given to the PUMA service mentioned several times in the preceding sections. Since PUMA is the evolution of an earlier project (dating to the year 2006) named Canone, *caserver* is a nickname referring to the Canone origins. As represented in Fig. 1 and Fig. 3, the client applications post their HTTPS requests to the Nginx web server. The latter forwards them to one or more *caserver* instances, working as a load balancer. The service is an object oriented, modular and plugin expandable C++ application relying on the *cumbia* and *cumbia-tango* [9] libraries to access the Tango control system in use at the Elettra synchrotron radiation facility. Once a request is received by the service, up to two actions take place:

- a synchronous reply is sent to the client immediately and normally carries either the result of the required operation enriched by additional information dependent on both the request and the underlying engine (e.g. from the Tango database) or an error message;
- if the operation is a subscription to the value of a source of data over time, updates are published on the channel the client subscribed to within the same request.

Clients, *caserver* and Nchan operate in a so called *pubsub* arrangement. The service publishes messages to channels using HTTP POST requests. Clients tune in to a channel to receive data through Server-Sent events.

The main type of requests that can be made to the server, herein named *methods*, are *read*, *conf*, *write* and *subscribe*. All of them are synchronous: an HTTP response is immediately sent back to the client. The subscribe method has the additional effect of data updates being delivered over a *pubsub* channel. A well behaved client shall pack together multiple requests into a single, more articulated, JSON string.

Engines and Modules

The service code is organised into modules. The Tango control system in use at Elettra is accessed by a dedicated module. This can be replaced (for example by an EPICS engine) and the *caserver* adapts to another control system.

A module has an interface to process messages received from clients and through *factories* several implementations of a module can be installed. Thus we can let the *reader module* use both the Tango and the EPICS specific *reader* implementation. A module is registered on the server and several ones can be installed with a given priority. As a consequence, a message from a client can be processed in sequence until one (or no) module satisfies the request.

New modules can be written and added to the service, although the preferred extension strategy is through the plugin system.

Cumbia

Cumbia activities use the standard C++ threads. Inspired by the Android *AsyncTask* interface [10] and paired with *cumbia timers* and *event loops* permit a multi threaded design of the service. Activities manage the transmission of data over the channels, the main socket server and plugins life cycles, the authorization process for write operations and so on. The *cumbia-tango* module is an abstraction layer facilitating the access to the namesake control system. Further implementation details are available on the project github page [11].

Plugins

The service features can be extended by plugins. Plugins are dynamically loaded from a specific folder and can be disabled as simply as deleting the corresponding object from the file system. The *caserver* provides *hooks* to which the plugins can register in order to receive specific pieces of information. For example, hooks related to readings notify plugins upon new data, subscribe and unsubscribe operations. Likewise, the read module can switch from an active to an inactive state, a message can be delivered synchronously on a socket or entrusted to a channel. All these events can be monitored by plugins. The service *supervisor* [12], mentioned in the section dedicated to reliability, queries a PostgreSQL database to retrieve data about the health of the *caserver* instances in execution. Such records are written by the *ca-db-plugin* [13], hooked to the service socket receiver state change and the periodic *heartbeat* events. Another plugin [14] offers *introspection* capabilities straight through HTTP representing in JSON (JavaScript Object Notation) format diverse operating conditions of the server at a given moment (number of activities, threads, readings, and so on).

THE SUPERVISOR

In the extent of the reliability of the PUMA service, and specifically for the accomplishment of the failover automation, the supervisor operates in conjunction with the *ca-db-plugin* discussed in the previous section. The

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

data recorded by the latter into a PostgreSQL database is analysed at regular intervals to determine whether every single *caserver* process is actually operating. Each instance is expected to register some kind of information periodically. If a unit fails, the supervisor undertakes recovery operations, redistributing the load formerly in charge of the missing service across the other processes online.

The architecture is designed so that if a policy of automatic and instantaneous restart of a broken down *caserver* is adopted, the restarted process regains control of its previous data *sources*[1]. The restart action must be undertaken promptly so that the supervisor is unaware of the failure.

The PostgreSQL database shall be centralised so that every instance of *caserver* can save its state regardless the host where it is executed.

Qt CLIENTS AND LIBRARY

Alongside platform independent web interfaces, Qt [15] native applications written in C++ can be developed on top of a library named *cumbia-http*. They are expected to work on all platforms supported by the Qt framework itself. More generally, any application written on top of *cumbia* / Qt is able to run transparently either connecting to the native control system engine[2] or to the PUMA service. The application does not need to be recompiled in order to select the available or desired engine at startup.

Cumbia-http

Cumbia-http is a *cumbia* module offered alongside engine specific ones such as *cumbia-tango* and *cumbia-epics*. Whilst the last two connect to the respective control systems natively and require complete access to the field and the software installation on the device as dependency, the HTTP module, connecting to the PUMA service, can be used by Qt applications from anywhere.

Cumbia Multi Engine Applications

A *cumbia* Qt application is unaware of the module used to read from *sources* and write to *targets*. The same software can run in the control room, where the native control system is thoroughly accessible, and at office or at home on a computer laptop, wherefrom the control system is unreachable and the native libraries are not installed. A *cumbia* multi engine application loads the desired module at runtime and offers the same user experience and native performance both from the control room and from home. From the developer point of view, a control panel can be designed, built and tested with utmost efficiency from anywhere using the *caserver* and then deployed natively on the field. Developers and end users alike treasured *cumbia* multi engine while working from home during the COVID-19 pandemic.

[1] Cumbia and QTango adopt the term “source” to name the quantity data comes from, for example, a Tango attribute. Recovery pertains to the context of readers, thus every reader has a “source” of data.
[2] For example, Tango or EPICS native installations.

WEB INTERFACES

Web interfaces are available on any device connected to the internet all over the world without any custom installation. Conversely, native applications benefit from having full knowledge of hardware. However, as soon as the quality and speed of web interfaces is close to native applications’, they are preferable because of their portability. An agnostic approach will implement all options letting the user choose the results rather than the technology. The same agnostic approach concerns the kind of devices supported; displays can be in a range about from 5 to 50 inches and can be touch or not; our goal is supporting as much devices as possible.

Mobile devices are usually strictly personal; this increases the opportunities of customization: apart from the style, the user can use predefined screens or create new ones arranging several items.

There is a particular screen called “starter”. Its purpose is to launch all other screens. Users can customize the starter screen either by a graphical tool or editing a JSON text.

Often the request of maintainability over years is in contrast with the choice of the most performing technologies available at the moment. So a compromise is necessary and the approach can be more aggressive or more conservative.

The choices made within the PUMA web interface development may be seen as conservative. We use JQuery [16] which enhances and simplifies JavaScript and has been quite stable over the past ten years. Bootstrap [17] is very helpful in being adaptive.

We had been using React for two years and decided to abandon it because in our particular use case the balance between the disadvantages (time consumed to keep the development environment updated) versus the advantages (modularity, state machine etc) was not satisfactory.

Adaptive Design

Adaptiveness is mainly provided by Bootstrap (Fig. 4) and by flexbox [18], which is a standard of CSS3 (Cascading Style Sheet) [19].

Within the PUMA framework a web designer is available to produce device interfaces or simple dashboards, all responsive because based on flexbox. Screens are saved in JSON format in a PostgreSQL database. Any saved screen can be used as a component in a new screen. JSON screens are interpreted consistently also by a previous app [1].

Advanced users can import in PUMA any HTML (HyperText Markup Language) and SVG [20] file. We consider "advanced users" those with a good knowledge of HTML, JavaScript/JQuery and CSS, also the rest of this section is addressed mainly to advanced users. Other readers, if interested in such details, can find plenty of explanations on the web. We suggest MDN (Mozilla Developer Network).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

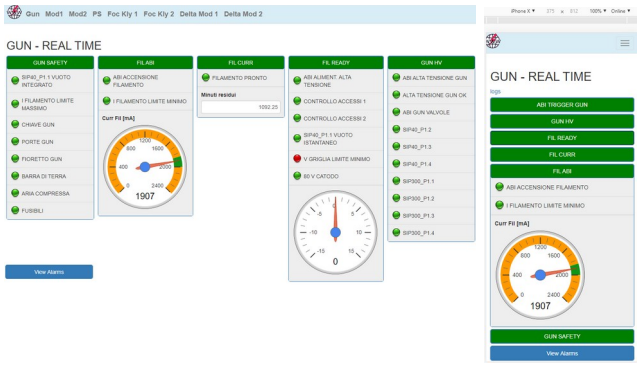


Figure 4: Desktop and mobile version of the same screen.

Vector Graphics

SVG is used to produce 2D graphics, in particular large machine synoptics. In this way a very good flexibility in interaction with the user can be obtained; two different JavaScript libraries allowing zooming in and out by using the mouse wheel or by pinching are used. While zooming or moving around, some contents are added, removed or changed. In particular some variables are subscribed and unsubscribed on pan or zoom change so that only the visible ones (and a few side buffers) are continuously updated (Fig. 5). The JavaScript function which subscribes and unsubscribes variables, is called at a maximum rate of 2 times per second.

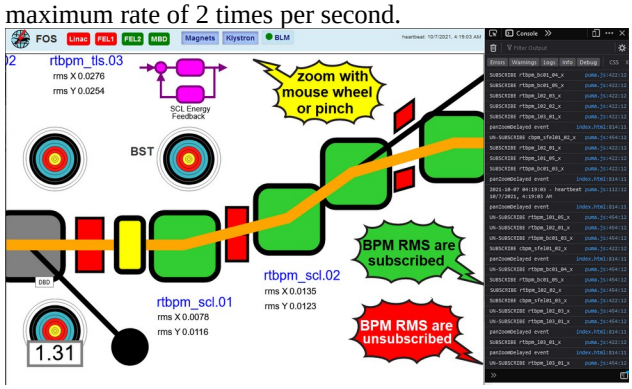


Figure 5: Some variables are subscribed and unsubscribed depending on the pan and zoom events.

In our working prototype this feature is completely transparent to the user, induced to believe that all variables are always updated without any loss in fluidity when moving the point of view. A demo is available on YouTube at the following URL: https://www.youtube.com/watch?v=z7FUDB7w2aw .

SVG is optimal for managing very complex 2D graphics and update any detail independently, but two main limitations subsist: it doesn't support 3D graphics and is not the best choice for displaying large quantities of data updated very quickly. WebGL [21] provides a very efficient solution to both problems.

PUMA provides an online HTML editor with a side preview which is constantly updated by the *keyup* event from the editor itself (Fig. 6).

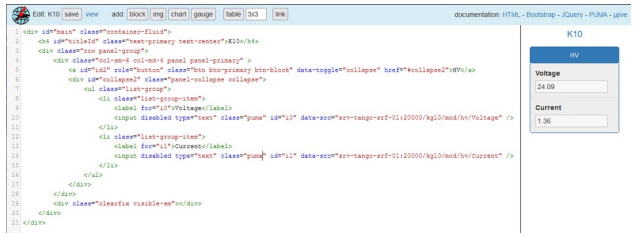


Figure 6: Web text editor with preview.

An authorized user can insert any HTML source which may include JavaScript, CSS and SVG. The connection to the control system can be implemented in any HTML or SVG tag by inserting a class "puma", a unique id and a custom data attribute [22] named "data-src". Only these steps are those necessary to connect to PUMA and there is no need to include any JavaScript. There is another optional attribute called "data-onupdate". Its value is the name of a JavaScript function that is triggered when new data is ready preventing the default update action. The "data-onupdate" JavaScript function is called with 3 parameters: the new value, the id of the calling tag (the same function can be used for more than one PUMA tag) and the source timestamp.

A one to one correspondence between tag and data-src is considered normal, though for exceptional cases this correspondence can be superseded. For example an array of booleans may be displayed as a table with green and red icons in the first column and labels in the second; such a table can be implemented using a hidden tag to associate a data-src to a data-onupdate function switching the colours of all icons. If a tag has to be updated according to the value of two or more variables, all variables should be associated to a different tag (hidden if not directly displayed) and to the same data-onupdate which provides a data synchronization mechanism and ultimately updates the multi-dependant tag.

The same screen can be reused in different contexts by inserting parameters into the data-src value.

SVG tags behave essentially in the same way as HTML tags; nevertheless there are a few differences to be taken cautiously in consideration, for example the JQuery [16] expressions .is(":hidden") and .is(":visible") work fine on HTML and SVG tags on Firefox, but they don't work on SVG tags on Chromium; instead the almost equivalent JQuery expression .css('display')=='none' works on SVG on all the most popular browsers (Fig 7).

It isn't very difficult to produce a thousand SVG elements image with a text editor, although a few graphical editors are available, for example Inkscape [23]. They tend to produce a lot of extra tags, that can be reduced by external tools which simplify the files.

In PUMA there are simplified versions of generic tools to browse all available control system variables, start and stop servers, monitor trends of a single variable. We also experimented with launching both web screens and desktop native applications from a browser. The second operation is explicitly forbidden by JavaScript. We overcame this limitation with a tiny bash script which executes the browser with an id as parameter, and waits in long polling mode [24] for a command to launch. Long

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR02

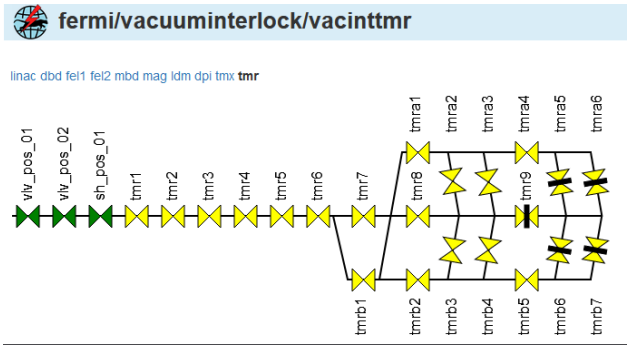


Figure 7: SVG on mobile.

polling is a cURL call to a particular page which receives an answer (resolves) only when there is a new event (a panel to be launched). Long polling is a technology much less advanced than WebSocket and SSE, but in this particular case it is still effective.

TEST PHASE

The PUMA service, the web interfaces and the *cumbia* HTTP module have been used constantly and efficiently during the whole COVID-19 pandemic. A dedicated phase addressing stress and failover tests is currently underway. Web and Qt clients have been written to investigate and diagnose limits and points of failure. Examination results shall be stored into a database so that several parameters, such as synchronous reply time, speed and resource usage under pressure, can be monitored in the long run and improved across version updates. Initial experiments prove that a failing service process is detected by the supervisor and its load is redistributed amongst the online instances of the cluster.

In order to evaluate the overall performance of PUMA two web based tests have been developed: the first one subscribes and unsubscribes a configurable number of variables per second, the second one simulates panning and zooming of a real screen. As a result we reached about 300000 subscriptions per hour for several hours.

CONCLUSIONS

The PUMA framework proved to be essential to develop and run graphical user interfaces from home during the COVID-19 pandemic, the other sole alternatives being either remote desktop solutions or X forwarding through SSH (Secure Shell). The first imply an amount of data compression degrading the graphics level of detail along with a moderately slow interaction, worsening to extremely sluggish in the second case. Actually, the authors' existing environment imposes two SSH tunnels over a VPN (Virtual Private Network).

The present day provides plentiful means in terms of computing speed and network bandwidth. Yet, they need to be economized. The data exchanged between the PUMA services and their clients involves only information relevant to the user: configuration and values from the control systems. In this sense, PUMA has an "ecological" approach: every bit is essential to knowledge, not even one carries redundant data and the user experience is always immediate because the resources of the device are operated natively. In case of portable devices, a remarkable battery saving shall be expected. Furthermore, due to the publisher – subscriber pattern over channels, the deployment of the PUMA framework in the control room on behalf of the native control system would definitely relieve the pressure on the devices and their servers. One need only think that umpteen readings of the same variable by PUMA clients translate into a single one to the control system.

The PUMA architecture has been planned with security, scalability and fail-safety in mind. On one side, the design contributes to administer the access and relieve the pressure on the control system. From the clients perspective, they are conferred scalable fast data exchange and lean on a failover redundant structure.

REFERENCES

[1] L. Zambon, A. I. Bogani, S. Cleva, E. Coghetto, F. Lauro, "Web and multi-platform mobile app at Elettra", in *Proc.* 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, paper TUSH103, pp. 984-988. doi: 10.18429/JACoW-ICALEPCS2017-TUSH103

[2] https://en.wikipedia.org/wiki/WebSocket

[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors

[4] https://en.wikipedia.org/wiki/Server-sent_events

[5] https://www.nchan.io

[6] https://www.nginx.com

[7] https://redis.io

[8] https://nchan.io/#high-availability

[9] https://github.com/ELETTRA-SincrotroneTrieste/cumbia-libs

[10] https://developer.android.com/reference/android/os/AsyncTask

[11] https://gitlab.elettra.eu/puma/server/canone3

[12] https://gitlab.elettra.eu/puma/server/ca-supervisor

[13] https://gitlab.elettra.eu/puma/server/ca3-db-plugin

[14] https://gitlab.elettra.eu/puma/server/ca-introspection-plugin

[15] https://www.qt.io/

[16] https://jquery.com/

[17] https://getbootstrap.com/

[18] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

[19] https://developer.mozilla.org/en-US/docs/Web/CSS

[20] https://developer.mozilla.org/en-US/docs/Web/SVG

[21] https://www.khronos.org/webgl/

[22] https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/data-*

[23] https://inkscape.org/

[24] https://en.wikipedia.org/wiki/Push_technology/

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR03

A MAJOR UPDATE OF WEB BASED DEVELOPMENT TOOLKIT FOR CONTROL SYSTEM OF LARGESCALE PHYSICS EXPERIMENT DEVICE

X. Xie, W. Zheng, M. Zhang, B. Rao, Y. Yang, F. Wu, Y. Jiang, P. Zhang, W. Wang, S. Li, International Joint Research Laboratory of Magnetic Confinement Fusion and Plasma Physics, State Key Laboratory of Advanced Electromagnetic Engineering and Technology, School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China

Abstract

The deployment of the control system called CODAC (Control, Data Access and Communications) is necessary for the operation of large-scale experimental facilities. CFET (Control system framework for experimental devices toolkit) is a flexible SCADA (supervisory control and data acquisition) software tool, which is used for the construction of a CODAC. CFET is fully based on open web technologies, it is easy to integrate all kinds of systems and devices into CFET. This paper has undergone a major iteration of CFET. HMI has been redesigned and implemented. The control engineer can use a web based WYSIWYG HMI editor to compose the HMI. In CFET, InfluxDB has been integrated. It is used to store the engineering data, and also visualize the data on the website. Docker based microservices architecture has been designed, putting CFET and dependent packages into a lightweight container. At present, CFET has been used in the CODAC system of J-TEXT tokamak and HUST Field-Reversed Configuration facility.

INTRODUCTION

For a long time, Experimental physics and industrial control system (EPICS) have been used to build large-scale experimental equipment control systems in the accelerator field. So far, the community has been very mature, and EPICS has very strong support for hardware equipment in the accelerator field [1-2]. ITER (as the world's largest tokamak) chose EPICS as the core framework of their control system [3-6], and chose EPICS Channel Access protocol as the communication protocol for the control network. Therefore, many other equipment in fusion field have chosen EPICS to build their own control systems. However, due to the differences in equipment and control requirements, EPICS has not demonstrated its advantages in the fusion community, and EPICS Channel Access protocol also has the problem of opacity and operability. Control system Framework for Experimental Devices Toolkit (CFET) framework is implemented as .NET standard libraries. It bases on Web technologies and uses HTTP as the control system communication protocol [7-8]. Web technology almost supported by all the devices. Countless web APIs have been published and consumed by all kinds of devices. It's also easy to integrate various new devices via the network, and provides flexible control system solutions for users in different scenarios. CFET is an efficient development tools with transparent protocol, and provides stronger support for new devices, also has the characteristics of easy integration and strong interoperability between subsystems.

This paper first briefly talked about the basic concepts of CFET. The third section will introduce the support of new communication protocol. Section 4 will introduce the redesigned HMI. The engineering data storage and management system will be described in the fifth section. CFET also uses docker to realize the cross-platform deployment of the user side rapidly. In the end, the application of CFET on HFRC will be briefly demonstrated.

BASIC CONCEPTS OF CFET

As mentioned before, CFET is a Web-based SCADA. Web plays a big role from websites (online services), online games, to smart sensor and IoT application. So CFET has good adaptability to a variety of different large-scale equipment, and can also meet new development models and new operating environments. HTTP is the most common communication protocol in web applications. The main communication module of CFET (CFET HTTP CM) uses the HTTP protocol of the RESTful architecture as the basic transmission protocol. The format of the RESTful framework is to use URI to locate resources and add verbs (intended action) in front of the resources. For example, if you need to know the switch status of light bulb A in the laboratory, the corresponding resource request should be "get + /lab/lightA/status". In HTTP CM, the client actions are mapped to HTTP verbs. There are three resource access actions, namely Get, Set and Invoke, mapping to HTTP Verb GET, PUT and POST. To different types of resources, HTTP verbs are different, otherwise an error that does not conform to the design principles will be reported.

The basis of interoperability is that every client in this system can understand each other conceptually, so we have to encapsulate the equipment in control system into a common model with consistent interfaces. The model has 5 types of resources: Thing, Status, Configuration, Method and event. Status, Configuration and Method are the property of Thing. A Thing can be either physical or logical. Event is a property based on publish/subscribe pattern. The subscriber can subscribe a resource and get notified on a certain condition.

NEW CFET COMMUNICATION MODULE

A CFET application, in principle, is allowed mount multiple Communication Modules. Each Communication

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Module corresponds to a protocol. When accessing resources, you can choose Communication Module via the protocol header, such as http://.

Due to increasing control requirements, CFET has developed some new communication modules: MQTT CM and WebSocket CM.

The HTTP protocol has a problem that only the client can send a request to the server, and the server return the query result. The HTTP protocol cannot enable the server to actively push information to the client. We can only use polling to get the current message if the server has continuous state changes. The biggest feature of WebSocket protocol is that the server can actively push information to the client, and the client can also actively send information to the server. CFET use WebSocket protocol to support remote events. WebSocket CM is integrated into HTTP CM, you don't need to change the protocol header, HTTP CM will automatically add 1 to its HTTP port number as the port number of WebSocket.

The MQTT (Message Queuing Telemetry transport) protocol is a protocol designed for communication between a large number of remote sensors and control devices that have limited computing power and work on a low-bandwidth, unreliable network. Provide one-to-many message publishing, uncoupling subscriber from publisher. MQTT has a wide range of applications in the Internet of Things, small devices, mobile applications, etc. CFET MQTT CM still uses RESTful architecture. The only difference from HTTP CM is that you need to replace "http://" in URL with "mqtt://" to select MQTT CM.

HUMAN MACHINE INTERFACE

Human machine interface (HMI) are developed separately and deployed on the operator's console in traditional control systems and always be a fixed interface on a limited platform. In web based control system, all HMI is a web site running on web servers. CFET provides a component called "WidgetUI". It is a WYSIWYG HMI editor, and the application developer can compose their own HMI via dragging and editing Widget. The component library of WidgetUI provide many widget: Label, Configurator, Gauge, Video player, Switch, Status light and so on. The following Fig. 1. Shows a simple HMI.

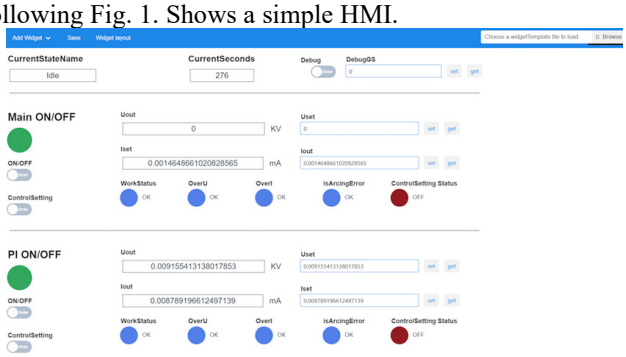


Figure 1: The HMI of high voltage power supply.

The completed interface can be directly saved as a Json file, which can be saved locally or in n a specified file path of CFET. The website can restore HMIs by parsing the Json file. And the website can also be accessed as a RESTful API. When a request come from a browser, the server would know that and return a web page to visualize the resource instead of a JSON object.

It is worth noting that, in addition to supporting access to CFET resources through the HTTP protocol, the website also supports subscribe a remote events through WebSocket protocol, which can be done by simply clicking on the website.

THE ENGINEERING DATA STORAGE AND MANAGEMENT SYSTEM

Engineering data refers to the non-experimental data generated during the experiment, such as sensor data, the voltage of the power and the temperature of the instrument, etc. Most of them have strong timing characteristics. Influxdb is purpose-built to handle the massive volumes and countless sources of time-stamped data produced by sensors, applications and infrastructure. CFET uses influxdb to store and monitor engineering data, which is capable of ingesting millions of data points per second. In CFET, we still write the control of Influxdb as a Thing with I/O interface exposed. Influxdb thing can be used with CFET Event to conveniently record any required engineering data.

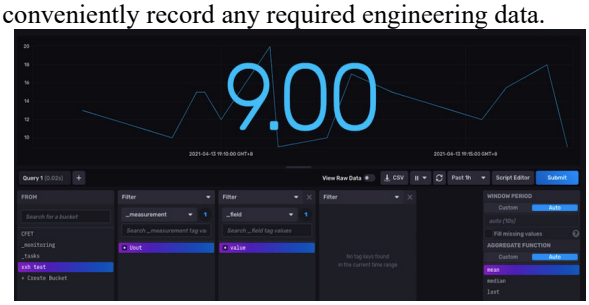


Figure 2: The interface of engineering data.

In addition to this, a dashboard interface (shown in Fig. 2.) and monitoring alarm interface built with Grafana are also used by CFET to visualize the engineering data.

APPLICATION ON HFRC

The Field-Reversed Configuration (FRC) device is a complex magnetic confinement fusion research device and is also a new type of fusion research device proposed in recent years [9]. The position and structure of this device are quite different from those of previous fusion devices such as tokamak, and can generate high-parameter plasma. HFRC is a FRC device being researched by Huazhong University of Science and Technology. HFRC contains multiple complex subsystems such as vacuum, power supply, and diagnosis. It is particularly important whether the control system can have the coordinated control capability of these complex and diverse subsystems, the real-time control capability of the discharge pulse, and the ability to provide flexible control functions to complete specific experimental content. Due to the flexibility and compatibility of CFET, HFRC chose CFET to build his control system. With the CFET framework, each subsystem can know any message from each other, and the central control system

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAR03

can coordinate and monitor by obtaining the status of each subsystem.

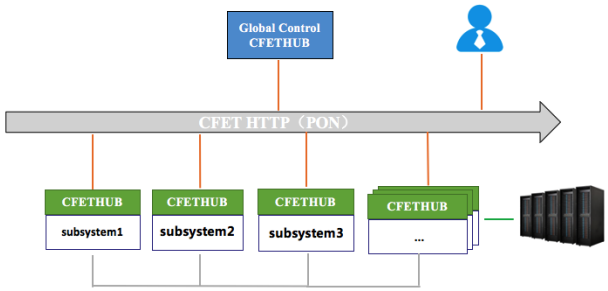


Figure 3:The Architecture of the control system of HFRC.

The HFRC control system is divided into two parts: the global control system and the sub-control system. The global control system coordinate subsystem to meet the control requirements of the overall operation of the device. The sub-control system consume these services by accessing CFET control system networks while exposing its own status API. The control system adopts a dynamic state machine mode to realize the process control of the discharge. Each subsystem can observe the state of the central control, execute its own actions and change its state according to the instructions of the central.

CONCLUSION

This work is aim for update web based development toolkit CFET. CFET bases on Web technologies and uses HTTP as the control system communication protocol to improve the interoperability of control systems. Although CFET has begun to take shape, there are still many functions that need to be improved. First this work added two new communication protocol module (WebSocket CM and MQTT CM) in CFET. Then the HMI was redesigned, and the component library was added. Users can access the control interface without restriction in any scene through the browser. CFET also be able to access Influxdb, integrates the storage and management system of engineering data, and users can also use the dashboard to monitor engineering data. Finally, the updated CFET has been applied to the HFRC control system, which meets the control requirements of HFRC daily experiments more flexibly and comprehensively, and also allows developers to develop the HFRC control system more efficiently.

ACKNOWLEDGMENTS

The authors wish to thank all the members in J-TEXT laboratory. This work is supported by the National Magnetic Confinement Fusion Science Program (No. 2017YFE0301803) and by the National Natural Science Foundation of China (No.51821005).

REFERENCES

[1] J. Zhang, M. Zhang, W. Zheng, G. Zhuang, and T. Ding, "Design and application of an EPICS compatible slow plant system controller in J-TEXT tokamak," *Fusion Engineering and Design,* vol. 89, pp. 604-607, 2014. doi:10.1016/j.fusengdes.2014.04.052

[2] L. R. Dalesio, M. A. Davidsaver, M. R. Kraimer, S. M. Hartman, K.-U. Kasemir, A. N. Johnson, *et al.*, "EPICS 7 Provides Major Enhancements to the EPICS Toolkit,"in 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS' 17), Barcelona, Spain, 2017, pp. 22-26. doi:10.18429/JACoW-ICALEPCS2017-MOBPL01

[3] A. Wallander, L. Abadie, H. Dave, F. Di Maio, H. K. Gulati, C. Hansalia, *et al.*, "ITER instrumentation and control—Status and plans," *Fusion Engineering and Design,* vol. 85, pp. 529-534, 2010/07/01/ 2010. doi: 10.1016/j.fusengdes.2010.01.011

[4] K. H. Kim, C. J. Ju, M. K. Kim, M. K. Park, J. W. Choi, M. C. Kyum, *et al.*, "The KSTAR integrated control system based on EPICS," *Fusion Engineering and Design,* vol. 81, pp. 1829-1833, 2006/07/01/ 2006. doi:10.1016/j.fusengdes.2006.04.026

[5] V. Vitale, C. Centioli, F. Di Maio, M. Napolitano, M. Panella, M. Rojo, *et al.*, "FTU toroidal magnet power supply slow control using ITER CODAC Core System," *Fusion Engineering and Design,* vol. 87, pp. 2012-2015, 2012. doi:10.1016/j.fusengdes.2012.05.006

[6] W. Zheng, M. Zhang, J. Zhang, G. Zhuang, Y. He, and T. Ding, "The J-TEXT CODAC system design and implementation," *Fusion Engineering and Design,* vol. 89, pp. 600-603, 2014. doi:10.1016/j.fusengdes.2014.03.048

[7] W. Zheng, Y. Wang, M. Zhang, F. Wu, N. Fu, S. Li, "Designing Control System for Large Experimental Devices Using Web Technology,"in 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS' 19), New York, USA, 2019, pp. 28-32. doi:10.18429/JACoW-ICALEPCS2019-MOBPP02

[8] W. Zheng, Y. Wang, M. Zhang, Z. Yang, and Y. Pan, "Designing CODAC system for tokamaks using web technology" *Fusion Engineering and Design, ,* vol. 146, pp. 2379-2383, 2019. doi:10.1016/j.fusengdes.2019.03.195

[9] H. Gota, M.W. Binderbauer, T. Tajima, S. Putvinski, M. Tuszewski, B. H. Deng, *et al.*, "Formation of hot, stable, long-lived field-reversed configuration plasmas on the C-2W device," *Nuclear Fusion,* vol. 59, pp. 112009, 2019. doi:10.1088/1741-4326/ab0be9

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

MACHINE LEARNING FOR ANOMALY DETECTION IN CONTINUOUS SIGNALS

A. A. Saoulis*, K.R.L. Baker, R. A. Burridge, S. Lilley, M. Romanovschi
ISIS Neutron and Muon Source, Didcot, UK

Abstract

High availability at accelerators such as the ISIS Neutron and Muon Source is a key operational goal, requiring rapid detection and response to anomalies within the accelerator's subsystems. While monitoring systems are in place for this purpose, they often require human expertise and intervention to operate effectively or are limited to predefined classes of anomaly. Machine learning (ML) has emerged as a valuable tool for automated anomaly detection in time series signal data. An ML pipeline suitable for anomaly detection in continuous signals is described, from labelling data for supervised ML algorithms to model selection and evaluation. These techniques are applied to detecting periods of temperature instability in the liquid methane moderator on ISIS Target Station 1. We demonstrate how this ML pipeline can be used to improve the speed and accuracy of detection of these anomalies.

INTRODUCTION

The ISIS Neutron and Muon source, located at the Rutherford Appleton Laboratory site in Oxfordshire, UK, creates neutron and muon beams used to perform a range of high quality scientific experiments. The facility has developed a great deal over its 35 years of operation [1], both increasing the complexity of the facility and the production of machine data. Currently, anomaly detection and response are generally handled manually by operators, and require large amounts of domain knowledge and expertise. Outlined in this paper is a pipeline to take unlabelled, continuous time series data and train a model that can detect anomalies on live data during operations, improving the response-time and effectiveness of reacting to these anomalies.

ISIS TS1 Methane Moderator

The facility accelerates protons up to 800 MeV, which are used to generate neutrons through a spallation process at ISIS Target Station 1 (TS1) [1], the first of the two target stations at ISIS. These neutrons are moderated at TS1 through several different moderators, one of which is a liquid methane moderator, in order to perform neutron scattering experiments. For high quality experiment data, very stable temperature in the methane moderator is required.

Whilst methane has many properties that give it excellent performance as a moderator [2], it is well known that it is susceptible to radiation damage [3]. The irradiation of the methane produces long chain polymers and releases hydrogen [3,4], the former causing the moderator to fail and require replacement roughly every six months. The build-up of free hydrogen within the moderator system causes loss of flow and leads to unpredictable pressure variability and spiking. This causes the temperature in the methane moderator to become unstable, which increases the variance of neutron energy leaving the moderator; it is therefore of key operational importance that these losses of flow are dealt with quickly. One method through which the operations team have dealt with this issue is through daily, scheduled "recoveries" of the moderator that consist of flushing through one third of the liquid methane in the system into a dump tank. This has improved its stability, but occasional periods of flow, and thus pressure and temperature variability, still occur.

The operations team currently have systems in place for detecting these instabilities, such as monitoring differential pressure in the system for any reduction in pressure. If an anomaly is detected, the operations team inspect the recent behaviour of the moderator and decide whether to run an unscheduled recovery in order to return the system to normal operation. The current systems often fail to flag up ongoing anomalies, leading to long periods of temperature variability that can cause the data recorded in downstream instruments to be unusable.

This paper will investigate the automated detection of these periods of temperature variability, with the goal of aiding the operations team to track down and fix issues faster. The paper will make use of supervised Machine Learning (ML) algorithms, which require labelled data (i.e. each data instance has an associated class label, such as whether it is "normal operation" or an "anomaly") to train a model. Finally, a brief description of the process of deployment of such trained models to live operation will be given.

DATA PIPELINE

Here, a pipeline is given that takes raw time series data without labels and produces a dataset that is suitable for training ML models. In the case of the ISIS TS1 methane moderator, there was neither a logbook nor a convenient signal that could be used to automatically label temperature anomalies in the historic data. One key contribution of this paper is to define a general procedure for automatically labelling periods of anomalous behaviour in historic time series data so that models can be trained to detect these periods during live operation. Note that while this paper focuses on a single signal (i.e. a univariate time series), these methods are generalisable to multivariate time series.

* alex.saoulis@stfc.ac.uk

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

Raw Data

The ISIS accelerator control systems have recently undergone several upgrades [1, 5] that have allowed for the long term storage of time series data from a huge number of sources across the facility. This has provided a wealth of raw time series data of signals across the accelerator side of the facility stretching back to the end of 2018. As is outlined in [5], this data is stored in a database called InfluxDB [6], which specialises in time series storage. This allows for fast and simple access to this time series data, greatly easing the use of this data for data analysis and ML purposes.

The raw time series data is collected at intervals of approximately 2 – 3 s, provided the value of a signal has changed. This leads to irregular and unaligned raw data that must be preprocessed before use in ML applications. The temperature data used in this paper is taken from signal *TC68M*, a thermocouple located close to the liquid methane.

Data Preprocessing

The first step of the data pipeline is to preprocess the data into a form that is practically useful and convenient for use in ML [7]. The preprocessing work was done in Python, primarily using the Python library pandas [8]. The data must be filtered such that only data covering operational periods of the accelerator is included in the training and test data. This is to ensure the training and test data matches the target domain data in production; in other words, the ML model should be trained and evaluated on the same type of data that it will see when it is detecting anomalies during live operation of the accelerator. This means that out-of-cycle periods, periods of bad/missing data (for example caused by a hardware malfunction), and periods outside normal operational bounds (such as temperature bounds beyond which it can be inferred that the moderator was not in use) must all be filtered from the data.

ML models require an input vector, also known as a feature vector, which must be of fixed length. For an irregularly sampled time series, simply using the raw data points as feature vectors would lead to temporally mismatched inputs to the model; a feature vector of length 100 could span from e.g. 3 minutes - 10 minutes, depending on how frequently the signal was changing during that period. Instead, feature vectors should be regularised such that each element of the feature vector corresponds to a fixed length of time. The raw temperature data is downsampled into bins of fixed time by taking a mean over that period. If there are no elements in a downsampled period, it is inferred that the temperature did not change from its previous value and the last known value is forward filled into the new bin.

The data must be scaled since the stability of many ML models require input data to be within reasonable bounds [7]. In the case of the TS1 methane moderator temperature, the mean of each temperature window was subtracted from the raw data. This ensured that the scaled values in the feature vectors always lay in the range of $[-10, 15]$, as well as helping to account for non-stationarity in the time series due to long-term drift of the moderator temperature. This choice was made for simplicity, and the conventional choice of z-scaling the data or normalising between $[0, 1]$ would also do.

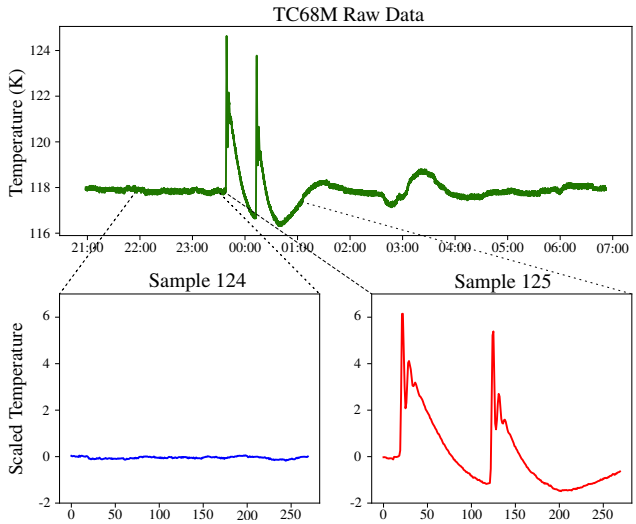


Figure 1: A short excerpt from the raw temperature time series, as well as two sample feature vectors generated from the data. Sample 124 is an example of normal operation, and sample 125 is an example of an anomaly.

Feature Vector Selection

In order to settle on a suitable feature vector, there are a few factors that must be taken into account. The window length must be long enough such that it can completely cover an anomaly event; this requires data exploration to look at the timescale over which anomalies occur in a system. A second important consideration is the bin lengths: these should be short enough to preserve any fine-grained structure in the time series that distinguishes normal operation from anomalies. Finally, in order to train models that can be evaluated constantly in real time, the feature vectors should be randomly sampled from the historic time series data such that a model can be applied at any moment in time

For the TS1 methane moderator, anomalies in the temperature tended to last between 1 – 2 hours, though certain indicators of anomalies such as rapid oscillatory periods occurred over much shorter timespans. As a result, 90 minute long feature vectors with bin lengths of 20 s were chosen, creating feature vectors of dimension (270, 1). The feature vectors were then generated by going through the downsampled time series data chronologically and extracting windows of length 270, skipping a small, random number of bins in between feature vectors to ensure that they had been randomly sampled. Roughly 4300 wholly distinct (i.e. no overlap) feature vectors were generated from the 2 years of historic temperature data, obeying the data preparation rules laid out above. Two such feature vectors, one from each class, are given in Fig. 1.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

LABELLING DATA

As mentioned earlier, in order to implement supervised ML algorithms, a class label corresponding to each feature vector is required. These are initially used to train a model to correctly identify the class of an input feature vector, and afterwards to evaluate the model on a test data set. Without any signals or logbooks that could be used to identify instances of anomalies in the historic data, labels must be generated directly from the feature vectors. This would typically be extremely time consuming and require a great deal of tedious manual work and expert knowledge. Here a generic method for partially automated labelling of time series feature vectors is presented.

Clustering through t-SNE

t-distributed Stochastic Neighbourhood Embedding (t-SNE) [9] is a dimensionality reduction technique, differing from the well known Principal Component Analysis (PCA) [10] in that it is a non-parametric, non-linear technique for embedding high dimensional feature vectors into low dimensional space according to their similarities. The main advantage of this technique over other dimensionality reduction methods is that it tends to produce well defined clusters of similar data points, which can be used to generate labels from feature vectors. Since t-SNE is tailored for visualising high dimensional data, it proves very well-suited to the data labelling process that will be introduced in this section.

There are also difficulties associated with using t-SNE; suitable hyperparameters for t-SNE must be chosen in order to generate meaningful clusters of data, which can become a tedious tuning process. Additionally, since the algorithm is stochastic, it may take several runs with the same hyperparameters to generate well defined clusters. One other disadvantage of t-SNE is that it is a non-parametric dimensionality reduction technique, meaning that, in contrast to e.g. PCA, the algorithm doesn't produce a mathematical formula that can map any high dimensional feature vector to its low dimensional embedding. This means that new data points cannot be embedded into the low dimensional space after the algorithm has been run.

In this paper, t-SNE will be used to generate clusters of similar looking feature vectors, which in turn will be used to generate labels for a large percentage of the data with minimal manual labelling. This relies on one key assumption, which is that a similarity score between feature vectors can be defined which is able to meaningfully distinguish between anomalous feature vectors and normal operation. The original t-SNE algorithm used the Euclidean distance between two feature vectors to calculate similarity, but for time series that can be very misaligned (due to the fact that for continuous time signals, there are no preset start and end points), a Euclidean distance will fail as a similarity metric. Instead, a distance metric better suited to misaligned time series is required.

Dynamic Time Warping

Dynamic Time Warping (DTW) [11] is an algorithm well suited to calculating the similarity between two misaligned time series. The algorithm is designed to work for time series that start and end at the same point, but whose main features vary in speed and so may be offset from each other along the time axis. There are a number of adaptations of DTW that can be used to tailor the algorithm for a specific use-case and set of constraints [12]. Even so, DTW works well even when there is no guarantee that two time series have the same start and end point; empirically, in preparation for this paper, standard DTW performed just as well as adaptations from [12] that accounted for these effects. As such, the standard DTW algorithm was used to calculate the similarity between feature vectors.

Note that in order to compare multivariate time series, the DTW algorithm can be applied to each individual signal (or a select few) across separate multivariate feature vectors, and a weighted sum of the individual DTW similarities can be used as the total similarity metric between two feature vectors.

Labelling the Temperature Windows

In order to apply t-SNE, the parallelisable function `metrics.pairwise_distances` from the Python library scikit-learn [13] was used to generate a similarity matrix explicitly between every pair of feature vectors, using DTW as the "metric". This is a computationally expensive step that has computational complexity $\mathcal{O}(n^2m^2)$, where n are the number of feature vectors in the data set, and m is the length of each feature vector. This step took around 30 minutes to run for the temperature samples on six cores on a desktop class machine. Note that this method may quickly become infeasible for very large datasets (e.g. when the number of samples $> 10^5 - 10^6$), or large feature vectors, without access to high performance computing clusters. Once the similarity matrix is computed, t-SNE can be run repeatedly for hyperparameter tuning without incurring the cost of calculating the similarity matrix each time. t-SNE embeddings of the temperature samples were created through a hyperparameter grid search until a suitable embedding was generated that had two distinct clusters of points. The final hyperparameters used for the scikit-learn function `manifold.TSNE` are given in Table 1.

Table 1: t-SNE Hyperparameters

Parameter name	Value
# Embedding Dims	2
Perplexity	30
Learning rate	25
# Iterations	5000
Method	"barnes_hut"
Angle	0.5

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

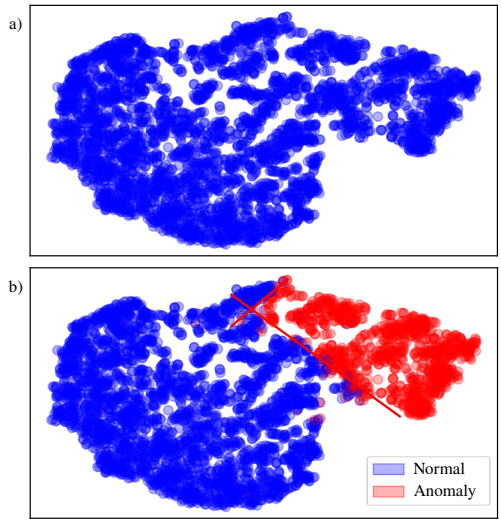


Figure 2: t-SNE plots showing the labelling process once two clusters have been generated. a) The raw 2D embedding generated by the t-SNE algorithm. b) The final labels assigned to the data after all labelling steps. The interactive plotting tool was used to get a first-pass estimate of the class boundaries, shown by the red lines.

In this case, as can be seen in Fig. 2, two large-scale, distinct clusters are easy to discern. An interactive plotting tool that allowed a user to select a point in the 2D embedding and view the corresponding time series was developed to inspect the data. Since the task was to classify normal operation versus anomaly, and not to distinguish between different types of anomaly, it was simple to manually define cutoff planes in the embedded space to make a first order approximation of the feature vector labels. If there are more than two classes, or the clusters are not simple to define manually, it could be preferable to use an unsupervised clustering algorithm such as k-means to calculate class boundaries. Then, the same plotting tool was used to manually change the label of any data points that were mislabelled on the first pass. This step can be time-consuming, especially in cases with large data sets and difficult-to-define class boundaries, and may need contributions from a domain expert to correctly classify edge-case time series.

One final step to ensure that this partially automated labelling process has generated accurate labels is to use a simple model or heuristic to classify the feature vectors, and compare the model classifications with the labels to check for bad labels. In this case, a rolling standard deviation of the time series was calculated, and feature vectors that exceeded a threshold were labelled as anomalies. This discovered a number of mislabelled data points, and was thus used to decrease the noisiness of the labels. Nonetheless, it should be borne in mind that no such labelling process will be perfect, and there will always be a degree of noisiness in the data set labels.

NEURAL NETWORK MODELS

Once a labelled dataset has been generated, a model can be trained using supervised ML. Neural networks have shown great promise for time series classification tasks [14], with recent papers and architectures delivering state-of-the-art performance on benchmark time series datasets [15]. This paper will introduce, train and evaluate three different neural network (NN) based models, as well as explore the differences between them. All of the models were created, trained and evaluated in Python using Keras [16] as the frontend for the popular ML library Tensorflow [17]. Neural networks have an added of bonus of generalising naturally to multivariate time series since they are data driven models.

Feedforward Neural Network

The simplest type of neural network is a feedforward neural network, also known as a multi-layer perceptron. The input to the neural network is the time series feature vector. Each subsequent "layer" of the neural network can be thought of as a matrix multiplication by matrix of size $m \times n$, where m is the size of the input vector and n is the number of "nodes" in a layer, followed by an elementwise non-linear transformation acting on the result of the matrix multiplication, known as an "activation function". The input is "fed forward" through the network of several layers, each of which can have different activation functions and numbers of nodes, until a final "output" layer that transforms the penultimate vector into a vector of length two, with each element corresponding to the probability that the feature vector belongs to either the "normal operation" class or "anomaly" class.

There are two main problems with using feedforward NNs for time series classification of the sort that has been described in this paper. Firstly, feedforward NNs learn the relationships between fixed elements in the input vector; since the position of features in the time series can occur at any point in the input, and their positions with respect to one another will be highly variable, a feedforward NN is not well suited to learning specific relationship between features. That said, for simple problems, it should still have some capacity to learn general features that it can use to make correct classifications. Secondly, feedforward NNs are generally thought of as black-box models, which means that it is very difficult to understand why a NN has made a particular classification. This can be problematic for models used in a control system to make decisions, since an operator may want to have some understanding of why a model detects an anomaly before making any changes.

Attention-based LSTM

The Long Short-Term Memory (LSTM) network has emerged as a valuable tool for time series classification and prediction problems in ML [15, 18]. It is an improvement of the Recurrent Neural Network (RNN) architecture, which processes an input vector in temporal order, preserving the causal structure of a time series that is generally lost in

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

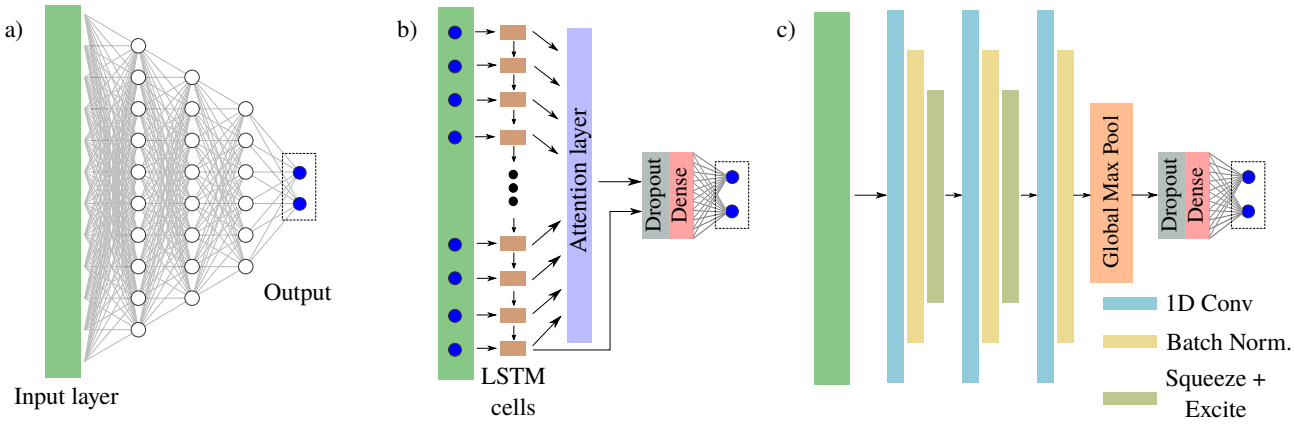


Figure 3: The three types of neural network used in this paper. ReLU activation function everywhere unless specified. Input layer always has dimension 270, output always has dimension 2 with softmax activation. a) The simple feedforward network with 4 hidden layers of size 64, 32, 16, 8 respectively. Dropout of 0.3 between each layer. b) The ALSTM architecture. Each LSTM cell had 16 nodes. The attention layer output and final LSTM output (both tanh activation functions) are combined before a dropout layer and a densely connected output layer. c) The CNN architecture with 3 1D convolutional layers: number of filters (32, 64, and 32) and filter size (8, 5, and 3) respectively. See [15] for a detailed explanation of architectures b) and c).

simpler NNs like feedforward networks. The LSTM introduces features that allow for information to be preserved throughout the network, giving it the ability to learn long-term temporal dependencies in the feature vectors.

This architecture is then expanded to include an attention mechanism, creating the Attention-based LSTM (ALSTM) [15]. In short, attention in neural networks allow the network to learn how to identify the most important areas in an input feature vector to make a classification, further expanding the ability of the LSTM to learn long term dependencies. Attention also has the bonus of providing some explainability to the neural network; after evaluating the neural network on an input feature vector, the attention activations can be examined to see which portions of a time series were most important in making the classification.

1D Convolutional Neural Network

The convolutional neural network (CNN) rose to prominence in the field of deep learning as a standout performer for image classification [19]. While it is most commonly used for tasks like computer vision, 1D CNNs have also been shown to perform well in time series classification tasks [14, 15]. CNNs work by learning filters that are run across an input feature vector to identify small scale edges and features, and then combining activations of these filters to build higher level features. The parallel between how CNNs work and how a human goes about time series classification are striking; much like a CNN, a human searches for shapes that match their understanding of what an anomaly might look like, and bases their classification on what types of shape appear in the time series.

Just as with the ALSTM, there has been a large amount of research that focuses on making CNNs explainable. These techniques include saliency maps [20], which use network activations to generate maps of regions of importance over the input, and occlusion methods [21] that mask regions of the input to determine which parts of the input were most used to make the classification decision.

TRAINING AND RESULTS

Each of these types of models were explored through network architecture tweaks and hyperparameter tuning. The final architectures and hyperparameters are given in Fig. 3. Since there was a data imbalance with a ratio of roughly $3:1$ between normal operation and anomaly in the feature vectors, anomalies were given a 3× class weight in the training process to ensure that the models would not learn a bias toward the more frequently occurring class. All of the models were trained using the default Keras Adam optimiser, and all models finished with a softmax layer to output a vector of probabilities of the input belonging to each class.

Of the 4300 time series samples, 65% of samples were used for training, 15% of samples were used for validation and the remaining 20% were used for a holdout test set on which each model was evaluated just once after the hyperparameter tuning was complete. The test set accuracies are the best indicator of out-of-sample performance, and are given in Table 2.

Table 2: Model Performance on the Test Set

Network Type	Accuracy (%)
Feedforward NN	97.5
ALSTM	96.0
CNN	98.3

The misclassifications on the test set were plotted and examined in each case. Since the ground truth labels were noisy due to the inherent problems in the data labelling

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

process, many of the "misclassifications" were actually the model outperforming the labelling process. In the case of the CNN, the majority of misclassifications were just bad labels, which suggested that the true CNN accuracy could be even greater than the accuracy given in Table 2.

APPLICATION

Once these models were trained and evaluated, the feed-forward NN and ALSTM (the CNN was omitted due to time limitations), as well as a simple heuristic that used the rolling standard deviation of the temperature time series to calculate the probability of anomaly, were deployed into production. To do this, a software stack using Docker [22] containers that sat on top of the existing ISIS accelerator controls MQTT messaging [23] system was developed. In the existing ISIS accelerator control system, live changes are sent to an MQTT broker, which are then consumed by several downstream applications. These messages were processed in the same way the training data was generated in the data processing pipeline to generate rolling feature vectors every 20 seconds. The feature vectors were then fed into the neural networks each time a new bin was generated (again, every 20 seconds). The outputs of the networks were then sent over the same MQTT network, after which they were collected by the metrics aggregator Telegraf [24]. The outputs, alongside the raw time series data, were stored in an InfluxDB [6] database, and displayed in real time in a Grafana [25] dashboard. The software stack is shown in Fig. 4.

Once the software stack and models had been tested for robustness and stability for a few weeks on the live system, a feature was added to generate control system alarms that would appear in the Main Control Room once an anomaly was detected. In order minimise the rate of false positives, an alarm was only generated if all models agreed with high probability that there was an anomaly for over 20 minutes. This alarm generation system was nominally successful, though since this work was done so near to the start of the ISIS long shutdown there was not enough time to generate data to report here.

FUTURE WORK

A natural extension of the work in this paper would be to extend the anomaly detection problem to one of anomaly prediction. This has clear practical importance to operators, since it would provide warning that could be used to minimise the impact of an anomaly, or potentially even prevent it entirely.

Some preliminary work was carried out using results given in this paper: the labels generated during the t-SNE labelling process were used to get approximate timestamps of the beginnings of the anomalies. Then, a number of extra signals recommended by domain experts (such as pressures, beam current, etc.) were collected and added to a feature vector directly preceding the anomaly, giving a multivariate time series acting as the input to the model. A model combining a CNN and ALSTM (see [15]) was trained to identify whether a time series was a precursor to an anomaly (or not), but it performed very poorly even after a long period of hyperparameter tuning. Some steps that will be explored in future work include feature engineering (likely after including an even larger set of time series signals), different ML models such as Gradient Boosted Trees, and addressing the large class imbalance through data augmentation.

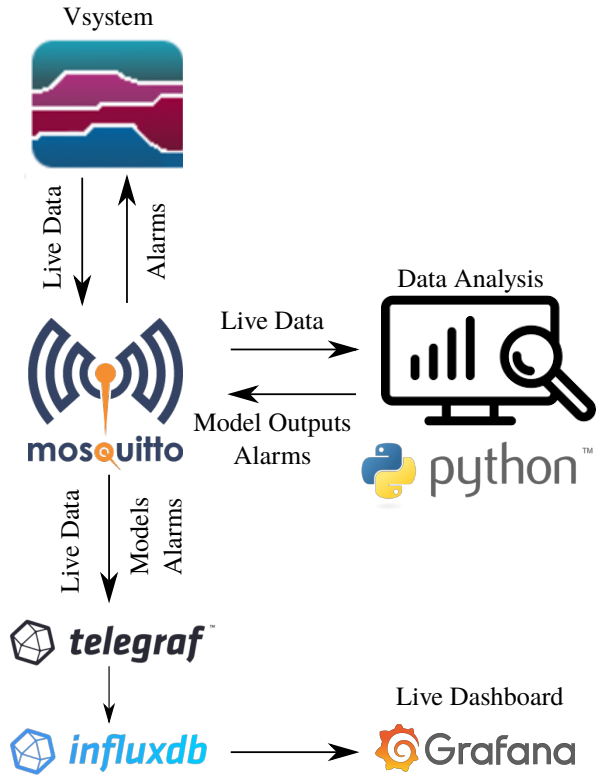


Figure 4: The software stack used to generate alarms and create live dashboards. The neural networks are evaluated in the data analysis layer, and their outputs are fed back into the MQTT messaging system.

CONCLUSION

A raw time series without labels was converted into a labelled data set using the clustering technique t-SNE. This dataset was then used to train several NN models which were first evaluated, and then deployed into live operations to generate alarms for the operations team any time an anomaly was detected. This pipeline provides a process through which anomaly detection can be applied to generic time series data streams, and proved a promising avenue for improving the response time to anomalies and thus availability of beam at large scale facilities.

REFERENCES

[1] Thomason, J.W.G., "The ISIS spallation neutron and muon source—The first thirty-three years" *Nuclear Instruments and Methods in Physics Research Section A: Accelerators,*

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Spectrometers, Detectors and Associated Equipment, vol. 917, pp. 61–67, 2019.

[2] Kirichek, O., Lawson, C.R., Draper, G.L., Jenkins, D.M., Haynes, D.J. and Lilley, S., "Solid methane moderators: Thermodynamics and chemistry", *Journal of Neutron Research*, (Preprint), pp. 1–6, 2020.

[3] Evans, D., "Irradiation effects in liquid methane used as a neutron moderator", *Cryogenics*, vol. 35, no. 11, pp. 763–766, 1995.

[4] Dean, R., Harrison, P., Burridge, R., Jenkins, D. and Probert, M., "Process filtration of liquid methane radiation products using centrifugal separation". in *Journal of Physics: Conference Series*, vol. 1021, no. 1, p. 012074, IOP Publishing, May, 2018.

[5] Finch, I.D. and Howells, G., "Controls Data Archiving at the ISIS Neutron and Muon Source for in-depth analysis and ML applications", presented at the 18th Int Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21), Shanghai, China, 2021, paper WEPV049, this conference.

[6] InfluxDB: time series database, `https://www.influxdata.com/`.

[7] Kotsiantis, S.B., Kanellopoulos, D. and Pintelas, P.E., "Data preprocessing for supervised leaning", *International journal of computer science*, vol. 1, no. 2, pp. 111–117, 2006.

[8] pandas: data analysis tool in Python, `https://pandas.pydata.org/`.

[9] Van der Maaten, L. and Hinton, G., "Visualizing data using t-SNE", *Journal of machine learning research*, vol. 9, no. 11, 2008.

[10] Jolliffe, I.T. and Cadima, J., "Principal component analysis: a review and recent developments", *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[11] Berndt, D.J. and Clifford, J., July. "Using dynamic time warping to find patterns in time series", in *KDD workshop*, vol. 10, no. 16, pp. 359–370, 1994.

[12] Giorgino, T., "Computing and visualizing dynamic time warping alignments in R: the dtw package", *Journal of statistical Software*, vol. 31, no. 7, pp. 1–24, 2009.

[13] Scikit-learn: machine learning in Python, `https://scikit-learn.org/stable/`.

[14] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.A., "Deep learning for time series classification: a review", *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[15] Karim, F., Majumdar, S., Darabi, H. and Harford, S., "Multivariate LSTM-FCNs for time series classification", *Neural Networks*, vol. 116, pp. 237–245, 2019.

[16] Keras: deep learning API for Tensorflow, `https://keras.io/`.

[17] Tensorflow: an end-to-end open source ML platform, `https://www.tensorflow.org/`.

[18] Siami-Namini, S., Tavakoli, N. and Namin, A.S., "A comparison of ARIMA and LSTM in forecasting time series", inn *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, pp. 1394–1401, Dec. 2018.

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[20] Simonyan, K., Vedaldi, A. and Zisserman, A., "Deep inside convolutional networks: Visualising image classification models and saliency maps", *arXiv preprint*, 2013. `arXiv:1312.6034`

[21] Fong, R.C. and Vedaldi, A., "Interpretable explanations of black boxes by meaningful perturbation", in *Proceedings of the IEEE international conference on computer vision*, pp. 3429–3437, 2017.

[22] Docker, `https://www.docker.com/`.

[23] MQTT: message broker, `https://mosquitto.org/`.

[24] Telegraf: metrics aggregator, `https://www.influxdata.com/time-series-platform/telegraf/`.

[25] Grafana: dashboarding tool, `https://grafana.com/`.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

A LITERATURE REVIEW OF THE EFFORTS MADE FOR EMPLOYING MACHINE LEARNING IN SYNCHROTRONS*

A. Khaleghi†[1], Z. Aghaei, K. Mahmoudi, H. Haedar, I. Imani, Computer Group, Imam Khomeini International University, Qazvin, Iran
M. Akbari, M. Jafarzadeh, F.A. Mehrabi, P. Navidpour, Iranian Light Source Facility Institute (ILSF) for Research in Fundamental Sciences (IPM), Tehran, Iran
[1]also at Iranian Light Source Facility Institute (ILSF) for Research in Fundamental Sciences (IPM), Tehran, Iran

Abstract

Using machine learning (ML) in various contexts is increasing due to advantages such as automation for everything, trends and pattern identification, highly error-prone, and continuous improvement. Even non-computer experts are trying to learn simple programming languages like Python to implement ML models on their data. Despite the growing trend towards ML, no study has reviewed the efforts made on using ML in synchrotrons to our knowledge. Therefore, we are examining the efforts made to use ML in synchrotrons to achieve benefits like stabilizing the photon beam without the need for manual calibrations of measures that can be achieved by reducing unwanted fluctuations in the widths of the electron beams that prevent experimental noises obscured measurements. Also, the challenges of using ML in synchrotrons and a short synthesis of the reviewed articles were provided. The paper can help related experts have a general familiarization regarding ML applications in synchrotrons and encourage the use of ML in various synchrotron practices. In future research, the aim will be to provide a more comprehensive synthesis with more details on how to use the ML in synchrotrons.

INTRODUCTION

Synchrotrons light sources are very large-scale experimental facilities. A synchrotron is a large machine whose size is about a football field (Fig. 1). In these facilities, electrons are accelerated to almost the speed of light. By deflecting electrons through magnetic fields, they create incredibly bright light. The electrons are deviated in the storage ring by different magnetic components such as bending magnets, undulators, wigglers, focusing magnets. This deviation results in a tangential emission of X-Rays by the electrons. The resulting X-rays are emitted as dozens of thin beams, each channeled down "beamlines" surrounding the storage ring in the experimental workstations where the light is used for research. Each beamline is designed for use with a specific technique or type of analysis [1]–[3]. The produced light is advancing research and development in fields as diverse as biosciences, medical research, environmental sciences, agriculture, minerals exploration, advanced materials, engineering, forensics [1]. The intense and highly focused light is used to study the dynamic and structure of materials down to atomic level using various techniques offered by different beamlines like diffraction, spectroscopy, tomography, and imaging [4]. Please see the references [1]–[3], [5] to see how a synchrotron works in more detail. Also, the list of light sources of the world can be found in [6].

Synchrotrons light sources worldwide are experiencing fast changes from traditional 3rd generation to multi-bend achromatic (MBA)-based 4th generation storage ring light sources to achieve high-brightness and low-emittance upgrades [7], [8]. The Advanced Photon Source (APS) and the Advanced Light Source (ALS) are both being upgraded

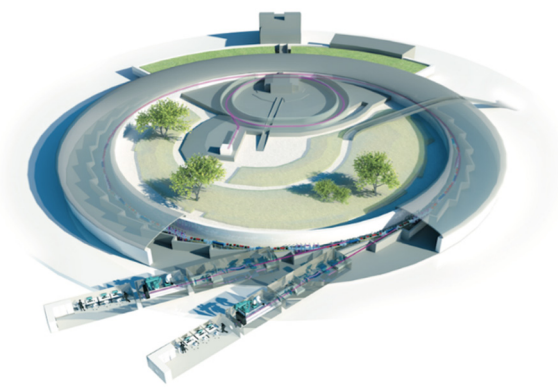


Figure 1: A 3D illustration of a synchrotron [2].

to MBA-based new rings. Diamond Light Source (DLS) designed a machine lattice based on double triple bend achromats [8]. The upgrades will substantially harness the light beam brightness from what is offered by the existing rings (the light brightness is much more greater than the sunlight) [7].

The rapid development of synchrotrons massively is accompanied by two significant challenges. First, the new rings drive for significantly lower emittances. Therefore, the beam dynamics in the rings become extremely nonlinear, causing smaller dynamic aperture and potentially smaller momentum aperture [7]. The extremely small emittance in a new ring needs much higher beam stability, which raises the need for a good understanding of the impact of environmental factors on the accelerator and

* Work supported by Iranian Light Source Facility (ILSF)
† khaleghiali@ipm.ir

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

beams [7]. Second, the speed of doing experiments and the amount of raw data collected during each experiment is increasing [9], [10]. Typically, each light source theoretically can produce one petabyte of data per day [11]. Or accelerators can generate three-petabyte data just in one experiment [12]. The manual analysis of such massive data volumes is no longer possible [9], [10].

Moreover, the lack of automatic data analysis prevents the delivery of new science from analyzing many collected raw data. These effects are collectively called "data deluge", which is a prevalent problem in synchrotrons [9]–[11]. To overcome issues such as data deluge, Machine Learning (ML), a subset of artificial intelligence that studies algorithms able to learn autonomously and directly from the existing datasets, can be very effective [13], [14]. Using ML in many contexts is increasing due to advantages such as automation for everything, trends and pattern identification, highly error-prone, and continuous improvement [15]. Even non-computer experts are trying to learn simple programming languages like Python to implement ML models on their data. Despite the growing practice towards ML, no study has reviewed the efforts made on using ML in synchrotrons to authors' knowledge. ML can be used to achieve benefits like:

- Reducing unwanted fluctuations in the widths of the electron beams produced at synchrotrons that, in turn, can prevent experimental noises obscured measurements. This work can stabilize the photon beam without the need for manual calibrations of measures [16].
- Preventing data deluge [9], [10]
- Reducing user-in-the-loop decision-making [9], [10], [17]
- Harnessing the brightness of light sources [17]
- Supporting efficient and clever monitoring and fault detection [17]
- Optimal setup using automatic alignment [17]
- Supporting stable conditions by providing real-time feedback to the experiments and end-users [17]
- Allowing users to focus on experiments at hand and best use the allocated beam time rather than manual setups and justifications [17]
- Providing instant and straight feedback from speedy physics-based simulations [17]

In this paper, we are focusing on efforts investigated ML applications in synchrotrons. Also, the challenges of using ML in synchrotrons and a short synthesis of the reviewed articles were provided. Our literature review can help related experts have a general familiarization regarding ML applications in synchrotrons and encourage the use of ML in various synchrotron practices.

MACHINE LEARNING AND ITS RELEVANT CONCEPTS

Machine learning is widely used to make predictions or decisions, a subfield of artificial intelligence and the process of making a mathematical model without being explicitly programmed and using sample data, famous as "training data". In other words, ML algorithms can learn to complete tasks using raw data [14], [18]. ML is usually applied for classification, regression, clustering, anomaly detection, dimensionality reduction, and reward maximization [13]. Generally speaking, ML techniques can be classified into three main categories, namely, supervised learning, unsupervised learning, and reinforcement learning (RL) [9]. Supervised learning is valuable when pairs of input and desired output are available. An algorithm can generalize the problem from the given structured data and predict unknown input. Unsupervised learning algorithms solve the tasks where only input data is available [19]. Recently, Reinforcement Learning (RL) has also attracted particular attention. RL is based on dynamic environment-agent interaction, similar to a Markov decision process [4], [20]. The agent starts an action on the environment, and the environment reacts to produce a reward, which the agent uses to learn how to enhance its subsequent actions. RL approach does not require a prepared data set consisting of input-output pairs since the agent learns by the continuous interaction with the environment, varying depending on the action and its dynamics [19]. Finally, semi-supervised learning is halfway between supervised and unsupervised learning. In this case, the algorithm is provided with both unlabeled and labeled data. This category is instrumental when available data are incomplete and to learn representations [21].

Among ML algorithms, clustering and deep learning (DL) are very popular. Regarding clustering is categorized as unsupervised learning (needs no labeled data). It groups data in some clusters. The similarity between the data within each cluster is maximum, and the dissimilarity between the data assigned to different clusters is minimum. Clustering algorithms are whether based on centroid research such as k-means or are density-based like DBSCAN. They see clusters as areas of high density separated by low density instead of determining the centroids. K-means is the simplest and most common clustering algorithm [19], and DBSCAN is robust against outliers. It can be applied to eliminate them on different stages of measurements and correction processes [22]. In recent years, deep learning has emerged as the leading class of ML algorithms, now almost synonymous with ML to the public. Deep learning uses neural networks (NNs) (Fig. 2) composed of hidden layers carrying out different operations to find and explore complex data's representations. It improves the performance of classifiers beyond that common ML algorithms offer, especially in the circumstances involving large datasets with high dimensions [23], [24].

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

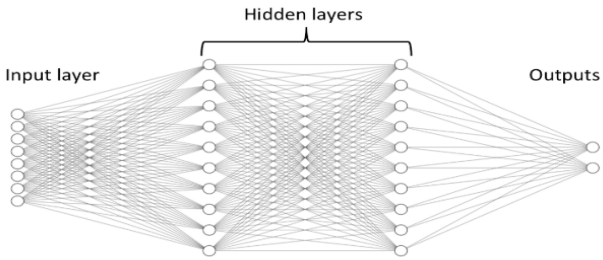


Figure 2: A fully connected NN with eight input features, two output labels, and two hidden layers including ten neurons each (Figure courtesy Alex LeNail).

CHALLENGES OF USING MACHINE LEARNING IN SYNCHROTRONS

Many AI/ML platforms for beamline tuning have been planned. However, most of them can not eventually be used regularly as part of an accelerator's central control system, mainly due to limitations such as required hardware, algorithms, software packages, and limited accessibility of large and suitable datasets [14]. The most critical challenges to scientists and end-users at synchrotrons for utilizing machine learning are [10]:

- the necessity for long-term data preservation and transfer, as well as the demand for data analysis pipelines
- The requirement for instant feedback helping on-site scientific and technical decisions during beamtime: Such feedback dramatically depends on the accuracy and automation implemented using sufficient hardware and software infrastructure for the real-time data evaluation and processing at synchrotrons.
- The need for user-friendly software packages to effectively control the extensive data generated in synchrotrons experiments: most beamlines users are not experts in data computing and management.

Alizadeh and Khaleghi also listed the most critical data management issues, which include: multiple source data, data analysis, data storage, data accessibility, data process, data format, data transfer, expensive data analysis tools, online processing, clustering, storage reliability, data mining, replication, real-time data collection and visualization [12].

Regarding these problems, a cross-domain and cross-facility solution that can accelerate creating a real-time user-friendly, advanced data processing platform at synchrotrons is needed [10]. The solutions must be versatile and flexible enough to be integrated at various experimental stations and cope with the heterogeneous requirements of different beamlines and experiments [25]. For example, supervised learning algorithms need advanced data management platforms because they require large amounts of reliable training data to construct reliable models [19], [26]. Unfortunately, while experimentalists have access to large datasets, these data are typically not tagged appropriately and thus are not suitable for supervised ML methods. Continual advances in hardware and software have enabled tremendous increases in the data collection rate. However, this boost in data throughput is not accompanied by a corresponding rise in identifying useful data and the value of each datum [27].

Some studies reported that a big data center which is termed a super facility is required for data management and processing [9], [10], [18]. Because the success of ML can be increased by the explosion in Big Data, advances in computational power, particularly the use of graphics processing units, and the development of more sophisticated ML techniques such as DL [18], [26].

In other words, these super facilities allow users to focus only on meaningful scientific data leading to discoveries and insights instead of dealing with unstructured and massive raw data. It can be achieved under users' simultaneous access to the synchrotrons' experiments with the designed real-time user-friendly platforms [9], [10]. In an interview conducted by Alizadeh and Khaleghi using ten light source facility members, it was concluded that 86% of participants were not familiar with big data [12]. Also, this study first classified synchrotrons experiments based on their techniques into three classes: imaging, scattering, and spectroscopy (Fig. 3). Then based on these main techniques, the researchers proposed a conceptual model for different data management aspects required for each method (Fig. 4). Some synchrotrons like the Shanghai Synchrotron Radiation Facility (SSRF) [9], the National Synchrotron Light Source II (NSLSII) [4], [17], and the Stanford Synchrotron Radiation Light Source (SSRL) [7] have made some efforts to provide the platforms needed for robust data analysis and management. The main work of these extensive platforms is that they should support a complete automated process for real-time and offline access for data management and computations to different operations of synchrotrons by providing sufficient hardware and software infrastructure [10].

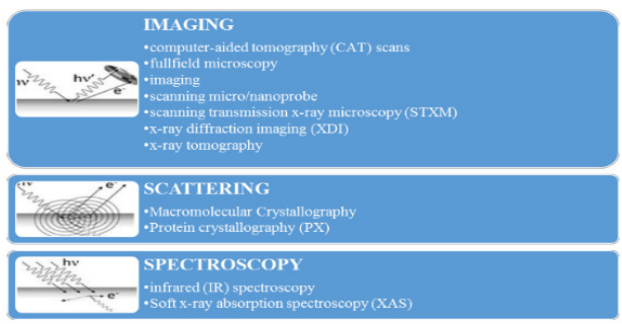


Figure 3: Synchrotrons experiments classification based on their techniques [12].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

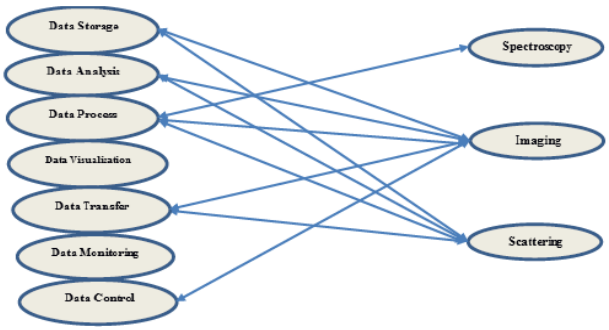


Figure 4: Different data management aspects required for each general synchrotrons experiments technique [12].

As the last point, we mention the concern that Hill et al. [22] noted that remote data analysis workflows and the use of distributed resources should be provided for ML purposes. The amount of beamtime required for a particular experiment reaches a limit as experiments become shorter. Therefore, it is no longer valuable for a user to visit the synchrotron physically. For this, remote data analysis and pipelines are essential in steering the experiment from a remote synchrotron and carrying out an optimal measurement.

THE USED/ DEVELOPED ML ALGORITHMS, SOFTWARE, AND PLATFORMS FOR DATA MANAGEMENT

High performance and low latency ML models are necessary to use ML techniques in everyday operations of synchrotrons [28]. Among these models, Neural Networks in synchrotrons have attracted particular attention due to their ability to facilitate image-centric big data science and many scientific imaging problems, such as denoising, feature segmentation, image restoration, and super-resolution [10], [29], [30]. Furthermore, NNs can explore nonlinear and dynamic behaviors [24].

NNs process the feature of image pattern rather than the value of each pixel, as with classical methods [31]. Moreover, image-based diagnostics can be used directly in accelerators both as outputs and inputs [14]. NNs' accuracy and efficiency for image recognition and classification have been proved from various applications [31]. Several tools have implemented NNs for synchrotrons operations. The Xlearn toolbox implemented NNs for multiple synchrotron X-ray imaging problems, which is an open-source Python package. The Features of Xlearn are [31]:

1- Correction of instrument and beam instability artifacts
2- Improving low-dose images
3- Feature extraction and segmentation
4- Super-resolution X-ray microscopy

The Xlearn can be easily integrated into existing computational pipelines available at various synchrotron facilities [32]. Moreover, the Xlearn is based on Keras [33] and Theano [34] packages. Keras is a well-known platform for neural networks and Theano for tensor flow computing. Keras and Theano also include GPU acceleration, the critical feature of applying NNs on extensive datasets. Please see [35] for source code, documentation, and information on contributing to this library. However, NNs are powerful in removing noise from reconstructed images. For training, they require collecting a dataset of paired noisy and high-quality measurements, which is a significant obstacle to their use in practice. In this regard, the Noise2Inverse was designed, a deep NN-based denoising method for linear image reconstruction algorithms that do not require any additional clean or noisy data. Recently, Hendriksen et al. [36] used the Noise2Inverse for deep denoising for multi-dimensional synchrotron X-ray tomography without high-quality reference data. This study applied the Noise2Inverse method to datasets acquired at two synchrotron beamlines. First, they used the technique on a static and a dynamic micro-tomography dataset from the TOMCAT beamline at the Swiss Light Source (SLS). Second, to investigate the possibility of accelerating the acquisition process using an X-ray diffraction tomography (XRD-CT), a dataset from the ID15A beamline at the European Synchrotron Radiation Facility (ESRF) was used. Results showed that Noise2Inverse is capable of accurate denoising and enables a substantial reduction in acquisition time while maintaining image quality. Liu et al. [30] introduced a deep NN model for real-time computed tomography at synchrotron light sources to improve the quality of tomographic reconstructions as data is collected. In turn, this method produces high-quality output more quickly and reduces the amount of data that must be collected. This method can be integrated into the real-time streaming tomography pipeline to enable better-quality images in the early stages of data acquisition. Using real-world datasets (tomography data, a common imaging modality at synchrotrons) collected at APS, results showed significant improvement in tomography image quality and system throughput.

Usually, different beamlines exit in a light source facility covering different scientific areas and utilizing different multi-dimensional detector technologies. Moreover, at synchrotrons, each beamline typically uses an individual streamline data acquisition software developed specifically for that beamline. The types of software are often incompatible with each other, making it difficult for scientists to compare data from different beamlines and other light sources. Therefore, AI/ML tools must also be compatible with these different beamlines [17], [37]. In this way, the NSLSII [17] developed the Bluesky Suite, a collection of Python libraries for data acquisition and management and mainly to tackle the data “variety” challenge and streaming and real-time data analysis at user facilities. There are capabilities like all data and metadata generated during an experiment can be emitted in real-time to other processes in the form of ‘documents’, Python dictionaries with comprehensible schema. The generated documents can be distributed locally or over a network. All beamline hardware is accessed via a library called ophyd. Or, the access to historical data is through an API called DataBroker.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

In another study recently published by NSLSII investigated reinforcement learning [4]. This paper demonstrated the use of RL methods for optimizing beamline operations. This study also explained how the Python-based Bluesky suite of data acquisition software enables RL applications at a beamline. This functionality by solving a classical RL problem, cartpole, in the Bluesky environment was demonstrated. Furthermore, the use of RL methods to address a prevalent scenario existed on high-throughput beamlines: maximizing data quality across multiple samples of different scattering strength within a limited time window. Finally, the challenges and overall strategy to realizing extensive development of RL methods at large-scale user facilities were discussed.

The European Organization for Nuclear Research (CERN) is the Large Hadron Collider (LHC) site, the largest and highest-energy particle collider globally [13]. The CERN has mainly focused on applying supervised and unsupervised ML techniques for various domains associated with beam dynamics studies. Some of these areas include beam commissioning of the collimation system, optimization of beam lifetime and losses, detection of collective beam instabilities, heating detection from pressure readings, and numerical simulations of dynamic aperture. For example, a fully automated software for beam commissioning of the collimation system using ML algorithms was developed. This new fully automatic alignment software was successfully used throughout 2018 in the LHC operations. Furthermore, this software will be used as the default software at the LHC in 2021. The time to align the collimators at injection was decreased by 71.4%, compared to the semi-automatic alignment, namely from 2.8 h to 50 minutes. Also, this tool was incorporated into the angular alignment implementation and successfully decreased the alignment time by 70%, requiring no human intervention. For a complete review of how ML techniques have been incorporated at CERN, please see [13].

Topaz3 is data manipulation and machine learning package implemented with python libraries for Macromolecular Crystallography (MX) at DLS. Specifically, it transforms electron density map data obtained from diffraction experiments and uses machine learning to estimate whether the original or inverse hand has clearer information. Tensorflow-gpu is required to use the machine learning side of Topaz3, which speeds up the training and use of neural networks [38].

One of the decades-old problems in synchrotron light sources facilities is that they simultaneously deliver light to dozens of beamlines. One side effect of this is that the movements of specific insertion devices (IDs), i.e., undulators and wigglers with variable magnetic fields, cause the electron beam's size to fluctuate. These fluctuations affect other beamlines' performance. Usually, changes are reduced using corrections based on a combination of static, predetermined physics models and lengthy calibration measurements. It is periodically repeated to counteract drift in the accelerator and instrumentation. Researchers at the ALS in Lawrence Berkeley National Laboratory showed that NNs algorithms can predict noisy fluctuations in the size of beams generated by synchrotron light sources. Therefore, they can correct changes before they occur (feed-forward vs. feed-back correction). Consequently, this approach significantly helps attain order-of-magnitude enhancement instability that fulfils the requirements for different light sources. For training the synchrotron, researchers fed electron-beam data from the ALS, including the positions of the IDs and blips in electron-beam performance raised by ID adjustments, into a NN. One key advantage of this approach is that the required data for retraining NNs can be obtained constantly, even while the feed-forward system is active during a regular user run [16]. Furthermore, continuous retraining allows the neural networks to continually adapt to a drifting machine and changes in ID configurations during run periods, independent of static physics models. The developed algorithm then learned the complex nonlinear relationships between the ID settings and vertical beam size and made corrections to negate the blips. NNs stabilized the vertical beam size at 0.2 μm or 0.4% of the beam size compared to 2–3% without correction [16].

Another light source facility that has heavily investigated using machine learning for light source facilities is the SSRL, a division of SLAC National Accelerator Laboratory, operated by Stanford University for the Department of Energy. A recent report which the SSRL published shows the research activities and achievements during the two-year R&D project of 'beam-based optimization and machine learning for synchrotrons' at SSRL [7]. R&D project was carried out in the development of machine learning techniques for synchrotron applications in three main areas: accelerator design optimization, beam-based optimization, and analysis of accelerator operation data. First, to implement online optimizations, they developed the Teeport platform. The control systems and programming environments on different machines may be different. Therefore, online optimization algorithms developed for one machine can not be easily applied to other systems. Usually, the optimizer is a Matlab script, and the evaluator is a Python script. Teeport decouples the algorithm implementation and the experimental systems by providing a universal middle layer that communicates between the optimizer and the evaluator. Therefore, they can communicate freely. In Teeport, a middleware between the evaluator and optimizer is inserted, acting as a data normalizer and signal forwarder. The data flows through the middleware. Therefore, to make the online optimization process more controllable and visible, one can add the control and monitor layers to the middleware (Fig. 5). The features of the platform include: online optimization experiment, fast switch between different optimization settings (the only necessary actions needed to switch between the different optimization settings concerning the code are switching the evaluator/optimizer id and/or update the configurations of

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

the evaluator/optimizer accordingly (Fig. 6)), optimization performance comparison, and optimization algorithm benchmark.

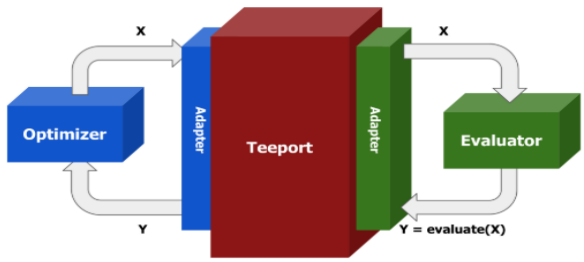


Figure 5: The architecture of Teeport. Teeport facilitates the application of optimization algorithms created in one programming environment to accelerators equipped with many different control systems and programming environments [7].

The Teeport platform can potentially become a centralized service for advanced optimization applications. It can be integrated lots of the optimization algorithms/test problems from various platforms, such as PyGMO, pymoo, PlatEMO, and Ocelot, to Teeport (Fig. 7). The R & D project also developed two ML-based global optimization algorithms for storage ring nonlinear beam dynamics optimization. The first is the multi-generation Gaussian process optimizer (MG-GPO) was used to solve multiple objective optimization problems. The second one is a neural network-based method to analyze accelerator operation data and study underlying environmental factors' impact on machine performance.

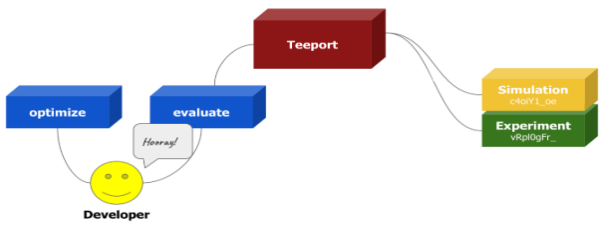


Figure 6: Fast switching between the simulation evaluator and the experimental evaluator [7].

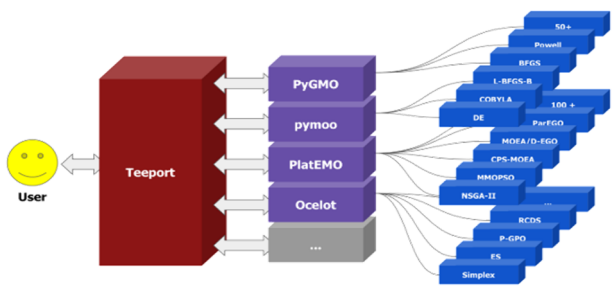


Figure 7: Teeport as a unified interface for the optimization algorithms [7].

BESSY II Light Source is another synchrotron that has started to use machine learning. The primary efforts made by this synchrotron described in [39] are: (1) beam lifetime can be successfully predicted in a time-series fashion using supervised learning models trained only with 185 accelerator variables readbacks, i.e., excluding previous lifetime measurements, and (2) the prototypes towards self-tuning of machine parameters in different optimization cases like injection efficiency and orbit correction using deep reinforcement learning agents have been implemented.

The Delta synchrotron for orbit correction used machine learning techniques [40]. Conventional Feed-Forward neural networks were trained on measured orbits to apply local and global beam position corrections to the 1.5–GeV storage ring DELTA. According to this study, it can be demonstrated that ML techniques are an alternative approach for automated orbit correction of the DELTA storage ring.

Fol et al. [19] focused on applying ML for beam diagnostics and incorporating ML concepts into accelerator problems. They identified four main areas that ML algorithms can be helpful, including virtual diagnostics, optimization and operation, beam optics correction, instrumentation fault detection. This study shows how different ML approaches can be incorporated for various functions of accelerators. For example, it has been concluded that reinforcement learning is suitable for solving complex control tasks. Or unsupervised learning is helpful for anomaly detection tasks such as detecting instrumentation defects, e.g., using clustering for faulty beam position monitors signal so that these methods can be performed directly without training in accelerator systems.

A SHORT SYNTHESIS OF THE REVIEWED ARTICLES

- To advance the field of incorporating ML in synchrotrons, a game-like project defining a reward scheme to train models to optimize a beamline efficiently is very effective.
- Good works have been done to facilitate data management and computing at synchrotrons, but they are ad-hoc based on different beamlines and synchrotron facilities. Therefore, future works can focus on converging these efforts in a seamlessly integrated platform for diverse beamlines with different requirements to provide impetus to employing ML in different operations of synchrotrons like accelerator design optimization, beam-based optimization, and analysis of accelerator operation data.
- Implementing remote data analysis workflows and the use of distributed resources are very effective for synchrotron practices. Because with the advances in synchrotron technologies, the amount of beamtimes needed for experiments is reduced tremendously, and an experiment can be done quickly. Therefore, physically visiting synchrotrons is no longer worthwhile.
- As evident from the above discussions, many researchers for developing ML models have used Python. There are three main reasons for this. First, for Python, such as HyperText Markup Language

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

(HTML), many existing codes are widely available for different use cases. By combining them, standalone platforms can be implemented more easily and rapidly. Considering such features, [14] suggested some people at accelerator laboratories can be determined to focus on the application of these tools to specific problems. Second, many libraries for ML purposes have been implemented in Python like Scikit-learn, TensorFlow, Keras, PyTorch, etc. Third, Python is very popular among non-computer experts due to its simplicity.

- Among existing ML algorithms and models, deep learning has been considered significantly. Almost, most of the reviewed articles have used NNs mainly due to their ability to explore nonlinear and dynamic behaviors and facilitate image-centric big data science.
- Unfortunately, on the Web, less publicly available good and large datasets are available for synchrotrons practices. It can mainly reduce the speed of using ML models in particle accelerators. By providing suitable datasets at cloud-based platforms like Kaggle in a short time, a variety of solutions can be provided for a given problem. For example, a successful competition was organized in 2014 by the high energy physics community, and it attracted over 1700 participants [14].
- Advanced online optimization using ML algorithms can support an efficient way of finding the ideal machine configuration.
- Combining big data with ML is already crucial. Storing, managing, and analyzing high-volume data are challenging problems that can be solved using this combination.
- In many contexts, usually, it is sufficient to use previously stored data for ML purposes. However, in synchrotrons, it is essential to develop online data streaming and management platforms because real-time usage of ML is vital for us in particle accelerators. Besides, processing and validating data after completing experiments lead to undetected problems and prevent online steering. Online ML algorithms and data processing platforms also can reduce the amount of data needed to be stored.
- The existing literature does not provide many direct comparisons between ML techniques using the same publicly available datasets. Therefore, to choose the best method that suits a given question, an empirical approach investigating different proposed ML methods on the same dataset is recommended.
- For storing data for ML techniques, mainly supervised algorithms, besides providing large datasets, it is essential to give techniques for tagging data and separating valuable and useful data from less or no useful data. This work reduces the amount of data needed to be stored and improves the efficiency of ML models.

CONCLUSION

The suitability of ML methods has been clearly shown in the performed review for beam energy, brightness, stability, etc. But much is expected from further application and extension of these approaches to the most diverse beamlines at synchrotrons globally, offering advanced capabilities, exploring the most varied, time-varying, and nonlinear relationships. There are many efforts to provide data management platforms for machine learning models and algorithms. However, these efforts are sparse and heterogeneous based on different beamlines and synchrotrons. It is beneficial to integrate them to propose a more efficient environment for incorporating machine learning in everyday synchrotron practices. Generally, in large experimental facilities such as synchrotron, neutron, and x-ray free-electron laser (XFEL), unifying ML-ready solutions is needed such that they should be general and transferrable to different beamlines and particle accelerators. Moreover, ML algorithms should also have the capability to be used online by providing online powerful data analysis platforms. Otherwise, some errors and anomalies may not be detected, and the amount of data that need to be stored will increase. Feed-forward correction, evaluation, and optimization (e.g., Feed-Forward Neural Networks) are more beneficial than feedback ones for many synchrotron practices, according to our review. In future research, the aim will be to present a more comprehensive synthesis with more details on how to use the ML in synchrotrons.

REFERENCES

[1] Australian Synchrotron, "What is a synchrotron?" [Online]. Available: http://archive.synchrotron.org.au/synchrotron-science/what-is-a-synchrotron [Accessed: 10-Oct-2021].

[2] ESRF, "What is a synchrotron?" [Online]. Available: https://www.esrf.fr/about/synchrotron-science/synchrotron [Accessed: 10-Oct-2021].

[3] ALBA, "What is a synchrotron?" [Online]. Available: https://intranet.cells.es/AboutUs/WhatIs [Accessed: 10-Oct-2021].

[4] P. M. Maffettone, J. K. Lynch, T. A. Caswell, C. E. Cook, S. I. Campbell, and D. Olds, "Gaming the beamlines—employing reinforcement learning to maximize scientific outcomes at large-scale user facilities," *Mach. Learn. Sci. Technol.*, vol. 2, no. 2, p. 025025, Jun. 2021.

[5] Lightsources.org, "Lightsources.org." [Online]. Available: https://lightsources.org/ [Accessed: 10-Oct-2021].

[6] Lightsources.org, "Synchrotron facilities." [Online]. Available: https://lightsources.org/lightsources-of-the-world/ [Accessed: 10-Oct-2021].

[7] X. Huang, M. Song, and Z. Zhang, "Report for Beam Based Optimization and Machine Learning for Synchrotrons at SSRL," 2021.

[8] T. Connolley, C. M. Beavers, and P. Chater, "High-Energy Adventures at Diamond Light Source," *Synchrotron Radiat. News*, vol. 33, no. 6, pp. 31–36, Dec. 2020.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[9] C. Wang *et al.*, "Deploying the Big Data Science Center at the Shanghai Synchrotron Radiation Facility: the first superfacility platform in China," *Mach. Learn. Sci. Technol.*, vol. 2, no. 3, p. 035003, Sep. 2021.

[10] C. Wang, U. Steiner, and A. Sepe, "Synchrotron Big Data Science," *Small*, vol. 14, no. 46, p. 1802291, Nov. 2018.

[11] PHYSICS TODAY, "Synchrotrons face a data deluge," 2020. [Online]. Available: https://physicstoday.scitation.org/do/10.1063/PT.6.2.20200925a/full/ [Accessed: 06-Sep-2021].

[12] S. Alizadeh and A. Khaleghi, "The Study of Big Data Tools Usages in Synchrotrons", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 1428-1431. doi:10.18429/JACoW-ICALEPCS2017-THPHA034

[13] P. Arpaia *et al.*, "Machine learning for beam dynamics studies at the CERN Large Hadron Collider," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 985, no. June 2020, p. 164652, Jan. 2021.

[14] S. National, M. Park, and D. Bowring, "Opportunities in Machine Learning for Particle Accelerators," 2018.

[15] R. Lund, "4 Benefits Of Machine Learning," 2021. [Online]. Available: https://techvera.com/4-benefits-of-machine-learning/ [Accessed: 09-Sep-2021].

[16] S. C. Leemann *et al.*, "Demonstration of Machine Learning-Based Model-Independent Stabilization of Source Properties in Synchrotron Light Sources," *Phys. Rev. Lett.*, vol. 123, no. 19, p. 194801, 2019.

[17] S. I. Campbell *et al.*, "Outlook for artificial intelligence and machine learning at the NSLS-II," *Mach. Learn. Sci. Technol.*, vol. 2, no. 1, p. 013001, Mar. 2021.

[18] M. Giovannozzi, E. Maclean, C. E. Montanari, G. Valentino, and F. F. Van der Veken, "Machine Learning Applied to the Analysis of Nonlinear Beam Dynamics Simulations for the CERN Large Hadron Collider and Its Luminosity Upgrade," *Information*, vol. 12, no. 2, p. 53, Jan. 2021.

[19] E. Fol, R. Tomás, G. Franchetti, and J. Coello de Portugal, "Application of machine learning to beam diagnostics," *Proc. 39th Int. Free. Laser Conf. (FEL'19)*, Hamburg, Germany, Aug. 2019, pp. 311-317. doi:10.18429/JACoW-FEL2019-WEB03

[20] R. A. Howard, *Dynamic Programming and Markov Processes*, 1 st editi. New York: Technology Press and Wiley, 1960.

[21] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," *npj Comput. Mater.*, vol. 5, no. 1, 2019.

[22] E. Fol, F. Carlier, A. G. Valdivieso, and R. Tomás, "Machine Learning Methods for Optics Measurements and Corrections At LHC," in *9th Int. Particle Accelerator Conf.(IPAC'18)*, Vancouver, Canada, Apr.-May 2018, pp. 1967-1970. doi:10.18429/JACoW-IPAC2018-WEPAF062

[23] A. L. Edelen, S. G. Biedron, B. E. Chase, D. Edstrom, S. V Milton, and P. Stabile, "Neural Networks for Modeling and Control of Particle Accelerators," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 2, pp. 878–897, Apr. 2016.

[24] P. S. Reel, S. Reel, E. Pearson, E. Trucco, and E. Jefferson, "Using machine learning approaches for multi-omics data analysis: A review," *Biotechnol. Adv.*, vol. 49, no. May, p. 107739, Jul. 2021.

[25] R. Gehrke, A. Kopmann, E. Wintersberger, and F. Beckmann, "The High Data Rate Processing and Analysis Initiative of the Helmholtz Association in Germany," *Synchrotron Radiat. News*, vol. 28, no. 2, pp. 36–42, Mar. 2015.

[26] D. H. Barrett and A. Haruna, "Artificial intelligence and machine learning for targeted energy storage solutions," *Curr. Opin. Electrochem.*, vol. 21, pp. 160–166, Jun. 2020.

[27] J. Hill *et al.*, "Future trends in synchrotron science at NSLS-II," *J. Phys. Condens. Matter*, vol. 32, no. 37, p. 374008, Sep. 2020.

[28] B. Blaiszik, K. Chard, R. Chard, I. Foster, and L. Ward, "Data automation at light sources," in *AIP Conference Proceedings*, 2019, vol. 2054, no. January, p. 020003.

[29] B. Wang *et al.*, "Deep learning for analysing synchrotron data streams," *2016 New York Sci. Data Summit, NYSDS 2016 - Proc.*, 2016.

[30] Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Deep Learning Accelerated Light Source Experiments," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, 2019, pp. 20–28.

[31] Argonne National Laboratory, "Xlearn," 2016. [Online]. Available: https://xlearn.readthedocs.io/en/latest/source/introduction.html

[32] X. Yang, F. De Carlo, C. Phatak, and D. Gürsoy, "A convolutional neural network approach to calibrating the rotation axis for X-ray computed tomography," *J. Synchrotron Radiat.*, vol. 24, no. 2, pp. 469–475, Mar. 2017.

[33] Keras-team, "keras." [Online]. Available: https://github.com/fchollet/keras.git [Accessed: 11-Sep-2021].

[34] Theano, "Theano." [Online]. Available: https://github.com/Theano/TheaNo [Accessed: 11-Sep-2021].

[35] "xlearn." [Online]. Available: https://github.com/tomography/xlearn [Accessed: 08-Sep-2021].

[36] A. A. Hendriksen *et al.*, "Deep denoising for multi-dimensional synchrotron X-ray tomography without high-quality reference data," *Sci. Rep.*, vol. 11, no. 1, p. 11895, Dec. 2021.

[37] S. Kossman, "Software Developed at Brookhaven Lab Could Advance Synchrotron Science Worldwide," *Brookhaven National Laboratory*, 2017. [Online]. Available: https://www.bnl.gov/newsroom/news.php?a=212470 [Accessed: 08-Sep-2021].

[38] Diamond Light Source, "Topaz3." [Online]. Available: https://github.com/DiamondLightSource/python-topaz3 [Accessed: 08-Sep-2021].

[39] L. V. Ramirez, T. Mertens, R. Mueller, J. Viefhaus, and G. Hartmann, "Adding Machine Learning to the Analysis and Optimization Toolsets at the Light Source BESSY II," in *Proc. ICALEPCS2019*, New York, NY, USA, Oct. 2019, paper TUCPL01, pp.754-760. doi:10.18429/JACoW-ICALEPCS2019-TUCPL01

[40] D. Schirmer, "Orbit Correction With Machine Learning Techniques at the Synchrotron Light Source DELTA," in Proc. *ICALEPCS2019*, New York, NY, USA, Oct. 2019, paper WEPHA138, pp. 1426–1430. doi:10.18429/JACoW-ICALEPCS2019-WEPHA138

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

RemoteVis: AN EFFICIENT LIBRARY FOR REMOTE VISUALIZATION OF LARGE VOLUMES USING NVIDIA INDEX

T. V. Spina*, D. A. D. Alnajjar, M. L. Bernardi, F. S. Furusato, E. X. Miqueles, A. Z. Peixinho
Brazilian Synchrotron Light Laboratory, CNPEM, Campinas, Brazil
A. Kuhn, M. Nienhaus, NVIDIA, Berlin, Germany

Abstract

Advancements in X-ray detector technology are increasing the amount of volumetric data available for material analysis in synchrotron light sources. Such developments are driving the creation of novel solutions to visualize large datasets both during and after image acquisition. Towards this end, we have devised a library called RemoteVis to allow the visualization of large volumes remotely in HPC nodes, using NVIDIA IndeX as the rendering backend. RemoteVis relies on RDMA-based data transfer to move large volumes from local HPC servers, possibly connected to X-ray detectors, to remote dedicated nodes containing multiple GPUs for distributed volume rendering. RemoteVis then injects the transferred data into IndeX for rendering. IndeX is a scalable software capable of using multiple nodes and GPUs to render large volumes in full resolution. As such, we have coupled RemoteVis with slurm to dynamically schedule one or multiple HPC nodes to render any given dataset. Remote-Vis was written in C/C++ and Python, providing an efficient API that requires only two functions to 1) start remote IndeX instances and 2) render regular volumes and point-cloud (diffraction) backend. In-memory data is transferred directly to dedicated servers over the network via *Remote Direct Memory Access* (RDMA), without involving temporary file transfers or centralized storage. NVIDIA IndeX is a scalable software for interactive visualization of large volumes in full resolution, written in CUDA and designed to render data leveraging multiple GPUs and/or multiple nodes of a distributed HPC environment. When the volume is received by an instance of IndeX, it is immediately injected for visualization by the user, who can interact with the volume in real time using a web viewer.

INTRODUCTION

Improvements in synchrotron light source technology are pushing forward the boundaries of X-ray microscopy imaging. Recently, 4th generation synchrotron light sources are increasing the amount of available beam flux and coherence, opening the doors to novel imaging techniques while representing an improvement of orders of magnitude with respect to previous generations. Hence, the entire imaging pipeline is evolving to make those powerful improvements available to the beamlines and their users; starting with the creation of fast X-ray detectors capable of acquiring frames at multiple kHz and reaching the development of high performance data processing, visualization, and analysis workflows.

In this paper, we propose a volumetric data visualization workflow in the form of a library called RemoteVis, to address some of the challenges imposed by the large data volumes being generated. RemoteVis is designed as an efficient C/C++ API for sending image volumes for remote 3D rendering, using NVIDIA IndeX [1] as the rendering

The combination of RemoteVis and NVIDIA IndeX aims to overcome several limitations of existing open source and commercial visualization softwares. For instance, Neuroglancer [2] is a community-supported tool for visualization of very large volumes originally created by Google. It is capable of displaying arbitrary (non axis-aligned) cross-sectional views of volumetric data, as well as 3D meshes and line-segment based models (skeletons). Neuroglancer uses a tiling mechanism to handle zooming with different resolutions of large volumes, which are displayed on the web browser. Napari [3] allows simplified visualization of n-D data via Python, while ImageJ/Fiji contains plugins for rendering volumes [4]. 3D Slicer [5] is a rich application devoted primarily to medical imaging, containing several visualization tools for these types of data.

Despite some of the advantages of the aforementioned softwares, they either make limited use of GPU capabilities for rendering 3D volumes, relying on a single device to do so, or only displaying cross-sections of data, while providing less than optimal techniques to handle larger volumes (i.e., data sets with more than 2048^3 voxels). Even commercial solutions, such as ORS DragonFly [6] and Thermo Fisher OpenInventor [7] (Avizo/Amira), tend focus on single GPU/single node rendering, with limited APIs that can be used to address the needs of modern synchrotron light sources. Finally, those softwares are usually implemented considering that data is essentially stored on disk.

We initially designed RemoteVis to tackle the issue of sending volumes generated on local servers connected to X-ray detectors to remote dedicated servers for visualization. The local servers are freed to receive data at high frame rates and to perform local processing on the data using custom multi-GPU code. Such processing may involve, for instance, frame correction operations and even high performance tomography reconstruction [8]. In parallel, the remote servers receive the resulting volumes and are responsible for rendering them at full resolution, with interactive responsiveness.

* This is a joint work between the following groups of the Brazilian Synchrotron Light Laboratory: Sirius Scientific Computing (SSC – TVS, AZP, MLB, and EXM), Throughput Enhanced Processing Unit (TEPUI – FSF), and Beamline Software (SOL – DADA); in collaboration with the NVIDIA IndeX team (MN and AK). Corresponding author: Eduardo Xavier Miqueles (eduardo.miqueles@lnls.br).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

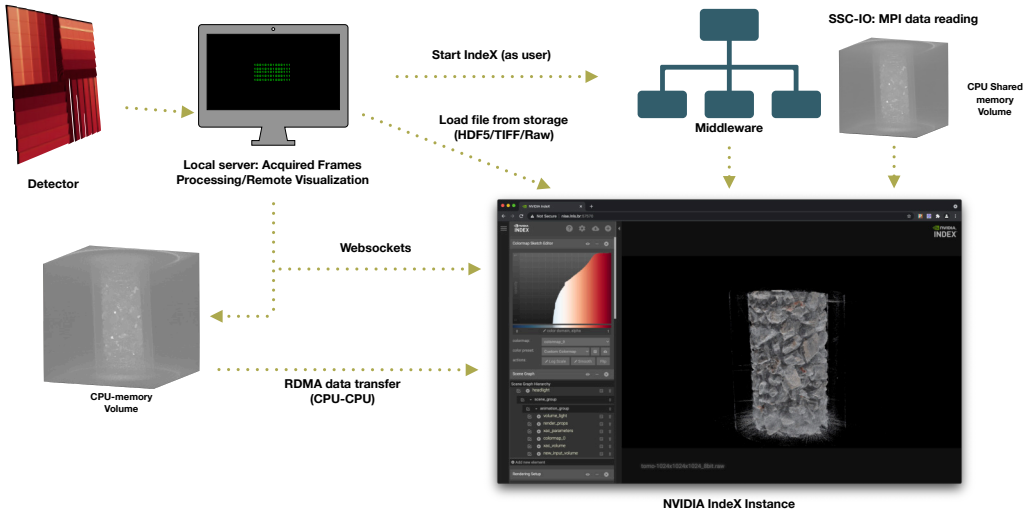


Figure 1: The proposed remote visualization workflow using the RemoteVis library to send volumes into NVIDIA IndeX for visualization, via RDMA data transfer.

The volume visualization solution we propose through RemoteVis provides much more than simple data transfer between nodes. We extended the library to serve as a platform for managing multiple instances of NVIDIA IndeX simultaneously, running in an HPC environment via job scheduling. Users of our APIs can transparently manipulate those instances with no more than a couple of C/C++ or Python functions calls. Furthermore, RemoteVis guarantees bidirectional data communication between NVIDIA IndeX and other softwares. This allowed us not only to send volumes, but also to retrieve portions of the volume that are being displayed for further processing. Moreover, NVIDIA IndeX provides two important functionalities that can be leveraged. First, the software implements the NVIDIA IndeX Accelerated Compute (XAC) facilities, through which custom functions can be created interactively for providing specific scientific visualization insights. Second, we are able to send pre-trained Deep Learning networks using RemoteVis to IndeX for inference. The software is then able to apply those networks instantaneously to the volume, thereby enhancing visualization via segmentation.

It is worth noting that Paraview [9] is an open source software that can use NVIDIA IndeX as a highly efficient rendering backend. We refrained from exploiting such integration to fully leverage the raw capabilities of IndeX in our workflows.

The paper is organized as follows. We first present the details about the RemoteVis library, including the capabilities of NVIDIA IndeX. We then briefly present some results before stating our concluding remarks.

THE REMOTEVIS LIBRARY

We may divide the RemoteVis library roughly into two major components. The first is an API written in C/C++ and Python whose primary goal is to perform data transfer/communication between NVIDIA IndeX and other softwares, which may be located in a same server or separated across an Ethernet network. The second is a middleware responsible for processing some of those API calls and for managing instances of IndeX, which may be running via a job scheduling mechanism such as slurm [10].

Figure 1 presents our visualization workflow combining RemoteVis and NVIDIA IndeX. Given a local server being operated by a beamline user for their experiment, the RemoteVis API is called to request a new IndeX visualization server instance. The RemoteVis middleware receives this request and initializes an instance on the beamline user's behalf in a remote server. All communication at this point is performed via TCP/IP socket transparently to the user of our API. While the NVIDIA IndeX instance is being initialized, the local server may be acquiring frames from the attached detector and/or conducting some processing on the received data, in order to produce the volume containing the sample of interest in CPU memory.

Once the sample volume is ready, yet another function call of RemoteVis performs two operations. First, the API asks the IndeX instance to prepare for receiving a new volume for data injection/rendering, using the software's websocket communication standard. Second, the volume is sent from the CPU memory of the local server directly to the CPU memory of the remote server running the IndeX instance via RDMA. When IndeX receives the volume, the software immediately injects the data into its rendering scheme by essentially transferring everything to multi-GPU memory.

RemoteVis is very flexible and its usage is not limited to transferring volumes from servers connected to detectors. Our API render volume function can be called with any 3D volume for remote rendering in CPU memory, regardless of the data origin. RemoteVis also provides another function to request the remote IndeX instance to directly load a file from

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

central storage. To further facilitate the use of RemoteVis by non-expert users, the library also incorporates a web interface through which the users can login and immediately start using an IndeX instance, without ever having to type a single line of code. In the next sections, we provide more details about NVIDIA IndeX and the components of RemoteVis.

NVIDIA IndeX

NVIDIA IndeX is a 3D volumetric interactive visualization platform that allows scientists and researchers to visualize and interact with massive data sets, make real-time modifications, and navigate to the most pertinent parts of the data, all in real-time, to gather better insights faster [1]. To achieve high efficiency and handle arbitrarily large data sets, IndeX divides a volume into smaller cubes and sends them for rendering across multiple GPUs and/or multiple nodes. IndeX uses a truly distributed database/compute environment to process the workload.

We implemented a plugin for data injection using IndeX's compute API, such that volumes are directly copied from CPU memory into the rendering pipeline for immediate visualization. Our plugin activates data injection after the RemoteVis library transfers the volume into the CPU memory of the server in which the IndeX instance is running, as will be detailed later.

The NVIDIA IndeX Accelerated Computing (XAC) interface enables scientists to rewrite the core data visualization routines at runtime. XAC-based programs are written in CUDA, compiled, and injected into the inner loops of the volume or surface raycasters to provide scientists with real-time visual feedback. We created XAC kernels for specific use cases of the Brazilian Synchrotron Light Laboratory and present some renditions in the results section (Figs. 2 and 3).

Visualization Server Middleware

NVIDIA IndeX ships with a web application built on top of the IndeX rendering library, consisting of a websockets server that receives JSON remote procedure calls from a React.js web viewer. The viewer displays the remotely rendered volume as an HTML5 video stream, with which the user interacts on their own web browser using mouse events. Those events are sent to the web server and effectively change the camera's position. We refer to this web application as a *visualization server instance* of NVIDIA IndeX in our work.

The RemoteVis library provides function calls to initialize, query, and shutdown instances of IndeX. The RemoteVis middleware is written in Python and processes those function calls, assigning a visualization server instance to the requesting user upon initialization. RemoteVis utilizes TCP/IP socket messaging between the user of our API and the middleware to ensure transparent client-server communication in a local Ethernet network. The visualization server instances may run directly on the server where the middleware resides as separate user processes or as scheduled slurm jobs, depending on the operation mode selected for the middleware. We detail later how slurm job integration works, but it is important to stress that regardless of the operation mode, the IndeX instance is started by the middleware as a process with system permissions of the requesting user. We thus ensure that all data access permissions are granted accordingly and respect the underlying HPC environment policies. Moreover, we assume the user has been previously authenticated into the local HPC environment and has been given the necessary privileges to use the RemoteVis API in their code.

The middleware starts the NVIDIA IndeX web server instance on a randomly chosen TCP/IP port that is then associated with the user. A URL is sent back through the API as the result of the initialization function for the user to access the IndeX instance on their web browser. The user may also invoke a query operation to retrieve the URL information later, or request the instance to be shut down programmatically.

Slurm Scheduling Integration

Since we propose RemoteVis as a framework for creating visualization workflows based on NVIDIA IndeX in an HPC environment, a common way of sharing resources in those settings is to make use of job schedulers such as slurm. When using the middleware in this operation mode, it starts visualization jobs via slurm and submits them to the job queues specified to run visualization server instances. Once again, those jobs receive the access permissions and slurm priority of the requesting user.

From the RemoteVis API perspective, the primary change in the slurm job operation mode is that the IndeX visualization server instance may not start immediately, depending on the state of the job queues. Hence, when the user requests an instance the middleware's default behavior is to create the slurm job and respond at once. Normally, the API function call is blocking, meaning that the user could readily send data for visualization once it returns, because the middleware waits for the instance to begin executing. This behavior changes when using the slurm backend, and the middleware returns the expected start date and time for the job instead of the URL. Currently, RemoteVis transfers the responsibility of verifying if the visualization job began to run to the user, who should use the API query function call to determine if the job has started.

When the IndeX instance job starts running, it communicates its current status to the middleware via RemoteVis. If the user queries the middleware at this point, the URL of the instance is returned as expected and data may be transferred to IndeX for visualization. It is worth noting that the NVIDIA IndeX job may be spawned on a server different than the machine in which the middleware resides, depending on the slurm configuration.

Since the user may not access the instance when the job begins its execution, possibly because the job started earlier than expected or the user forgot to check, the middleware automatically sends a shutdown command within a configurable grace period (usually 60 minutes) to release the computing resources for other jobs. The middleware also sends an estimated number of GPUs to slurm that are nec-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

essary for rendering the volume, based on the size and data types requested by the user in the initialization function call. Those parameters are enforced during data transfer calls.

SSC-RDMA: RDMA-based Data Transfer Library

RemoteVis implements data transfer to and from NVIDIA IndeX using RDMA. RDMA is a low latency protocol for reading and writing information stored on CPU memory remotely, bypassing the CPU to achieve high transfer rates. It was originally proposed for Infiniband networks, but has since been introduced to Ethernet through RDMA over Converged Ethernet (RoCE).

We created a standalone C library called SSC-RDMA (Sirius Scientific Computing RDMA) to achieve high data transfer rates using RDMA, based on [11]. SSC-RDMA's primary goal is to transfer image volumes. Hence, the library provides two high level functions that can be called in the receiver and sender sides, respectively. Both functions accept two `char` arrays that are used to transfer the volume of interest with arbitrary size and a header containing information to describe the volume. We represent a 3D volume in C array format (column-major) and therefore cast any type of volume to a `char` array for generic data transfer. The size of the linearized in bytes array must be shared between sender and receiver out-of-band, to ensure that both sides allocate the data with the proper size before the transfer takes place. The `char` header array assumes a fixed size of 128 KB and carries application-dependent encoded information about the volume (e.g., dimensions, data type).

When sending a volume to the visualization server instance for rendering, the C/C++-level RemoteVis `render_volume` function accepts as input a `char` array containing the volume data, the volume information (size and data type), and the information about the instance obtained during initialization or query. The RemoteVis `render_volume` function then sends a websocket message to IndeX to have it prepare for receiving a new volume with the corresponding size in bytes. Simultaneously, the SSC-RDMA data transfer `ssc_send` function is invoked by the RemoteVis `render_volume` call and waits for our IndeX plugin to connect for data transfer using the `receive_volume` function, from the RemoteVis API. The latter uses in turn the generic `ssc_receive` function of SSC-RDMA and data is finally transferred from the local server to the remote server where IndeX is running. Afterwards, our plugin immediately asks the IndeX library to render the volume by injecting it from the CPU to the GPU memory.

To retrieve a volume from NVIDIA IndeX, the process is similar to the above. The exception is that the `retrieve_volume` function from RemoteVis requests IndeX to send the volume that is being rendered via SSC-RDMA.

Python API and Jupyter

RemoteVis provides full access to the underlying C/C++ functions from Python, which in fact are extremely similar. The primary difference between the functions is that the Python version accepts Numpy arrays holding the volumes for remote rendering. Our API can be be called directly from Jupyter notebooks and include a special function of RemoteVis that returns a screenshot of the current volume visualization using IndeX an RGBA numpy array.

Web Client for User-friendly IndeX Access

Even though the RemoteVis API is very simple to use, many users are not interested in coding, or don't know how to code, an application. We provide a user-friendly web client through which the user logs into the system using their credential of the HPC system, and then the RemoteVis API is called to start an instance on their behalf.

Our web client communicates with a web server written in Python/Django, which validates the user credentials on the HPC system and calls the RemoteVis IndeX initialization function upon success. Since the RemoteVis API is used throughout the process, all IndeX access is unified through the RemoteVis middleware, thereby providing great flexibility for designing visualization workflows.

We have further expanded the original React.js IndeX web interface to allow users to load volumes from disk. The user may navigate on a centralized repository file system using a custom React.js component, select a volume for loading and pass the corresponding string to our IndeX plugin. That volume is read from disk and sent to IndeX for rendering.

SSC-IO: MPI-based Efficient HDF5 File Reading and Writing

Since many synchrotron light sources have been adopting the HDF5 file format for storing various imaging formats, we implemented a library to efficiently read and write volumes into memory. This library is called SSC-IO and it was motivated by the fact that the supported way of reading and writing HDF5 files in parallel requires using the Message Passing Interface (MPI) standard. MPI has certain limitations that hinder its use with IndeX. In particular, IndeX implements an efficient pipeline for reading volumes from disk using parallel callbacks that are issued only for the visible portions of the volume. Those callbacks are issued by separate threads and work well for raw volumes. However, the HDF5 file format does not provide proper parallelization at the thread level, thereby decreasing performance

We designed SSC-IO to read volumes from HDF5 using MPI and storing the result in shared CPU memory. This is necessary because MPI spawns processes instead of threads to achieve parallelization. Once the volume is read, we implemented a custom loading callback plugin for IndeX that reads data directly from CPU memory. SSC-IO parallelizes writing a volume from CPU memory to disk in the same way. We designed the library in Python such that we can easily extend it to multiple HDF5 file structures.

RESULTS

In the next sections, we present some results obtained using RemoteVis and IndeX. In particular, we focus on briefly

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

evaluating RDMA data transfer and detailing some of the rendering features of IndeX.

RDMA-based Data Transfer

Table 1 presents some data transfer results using SSC-RDMA. We conducted 12 experiments using three servers, one IBM Power 9 and two NVIDIA DGX-A100s. The Power 9 is connected to the DGX-A100 through a high speed Ethernet switch at 50 Gb/s. The connection between the DGXs is via 100 Gb/s Ethernet. The results in the table are the average of 5 executions using float 32-bit volumes with 4 different sizes. As can be seen, SSC-RDMA is very efficient and requires roughly 20s to transfer over 100 GB of data (for volume with size 3072^3 voxels). The selected sizes reflect some typical volume sizes produced by fast detectors in a matter of seconds. As can be noted, SSC-RDMA achieves roughly 45 Gb/s data transfer rates, indicating that there is still room for improvement in our code, even though we are able to move large volumes without using centralized storage.

Table 1: RDMA-based data transfer times using SSC-RDMA. Three settings were tested between three different servers, one IBM Power 9 and two NVIDIA DGX-A100. The IBM Power 9 ($P9_{50}$) is connected at 50 Gb/s to the NVIDIA DGX-A100 while the connection between the DGX servers is at 100 Gb/s. The selected volumes are of type float 32 bits (4 bytes per voxel) and vary in number of voxels. All times are in seconds. For DGX to DGX, the destination server was the same as $P9_{50}$ to DGX.

# vx	$P9_{50}$ to DGX	DGX to $P9_{50}$	DGX to DGX
1024^3	0.87 ± 0.01	1.13 ± 0.20	0.86 ± 1.48
1536^3	2.58 ± 0.09	3.00 ± 0.65	2.53 ± 0.47
2048^3	5.99 ± 0.09	7.17 ± 1.47	5.98 ± 1.12
3072^3	20.30 ± 0.26	21.89 ± 4.90	19.16 ± 3.66

XAC Rendering Kernels

We have implemented three different rendering techniques using XAC based on some use cases of LNLS. The first XAC kernel (S1) performs simple shading using local information such as gradient calculation on the volume. The second kernel (S2) simulates single scatter lighting and requires some extra memory for rendering, but produces shadow effects that improve visualization. The third is the most computationally demanding and is based on the ambient occlusion algorithm [12] (S3), requiring pre-computation to generate an auxiliary volume to determine shadow values. Note that all computation was done directly inside IndeX using the XAC technology.

Figures 2 and 3 presents two samples of X-ray computed tomography imaged at LNLS and redered using XAC. The first figure compares basic rendering with (S1) and (S3), while the second compares (S1) and (S2). The original volume of the silica bead experiment (Fig. 3) can be seen segmented in Fig. 4.

NVIDIA TensorRT CNN Inference

As previously stated, NVIDIA IndeX provides integration with TensorFlow/NVIDIA TensorRT such that pre-trained deep neural networks may be used to perform on-demand inference on the visible portion of the volume. We have devised softwares at LNLS capable of producing NVIDIA TensorRT-optimized inference engines from CNNs created for segmenting volumes, which can then be used with IndeX to provide advanced data interpretation.

Figure 4 presents an example of those results. In that case, we trained a 2D U-net [13] model to segment the phases of a silica bead fluid flow experiment containing four phases of interest (beads, water, gas, and background) using our custom softwares and then sent the resulting TensorRT engine for IndeX using RemoteVis. In that case, RemoteVis provides yet another function capable of taking a serialized engine and sending it to IndeX via SSC-RDMA for inference.

Point Cloud Rendering

NVIDIA IndeX was designed not only to render regular volumes, but also irregular data such as meshes and particle volumes (i.e., point clouds). We have incorporated into RemoteVis the capability of rendering point clouds submitted by the user similarly to how it is done for regular volumes. The corresponding function call accepts an array holding the `XYZ` coordinates of the points and another with the values of each one. Then, RemoteVis invokes RDMA data transfer for each of those arrays into the server where the IndeX instance is running and requests the software to render the point cloud as a particle volume. The points are rendered as spheres with radii proportional to the point values. The API is accessible from C/C++ and Python as always. Figure 5 depicts an example of point clouds being rendered from data obtained for 3D reciprocal space mapping, a common experiment in synchrotron light sources.

CONCLUSIONS

We presented the RemoteVis library as a framework for creating 3D visualization workflows using NVIDIA IndeX as the volume rendering backend. RemoteVis provides a data transfer mechanism via RDMA to allow volumes to be transferred from local servers to remote servers dedicated to visualization. The data in local servers may be acquired from X-ray detectors or simply produced by some other procedure. RemoteVis provides auxiliary functions that can load HDF5 files directly into IndeX very efficiently using MPI. Also, our library allows portions of the volume being rendered to be retrieved from IndeX into other softwares for further processing.

We also exploit in RemoteVis very useful techniques implemented in IndeX. First, we are able to create insightful rendering functions using the IndeX Accelerated Compute technique to provide advanced visualization for certain scientific cases. Second, we are able not only to render regular volumes but also point clouds int IndeX using RemoteVis, also via RDMA data transfer. Finally, we incorporate NVIDIA

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

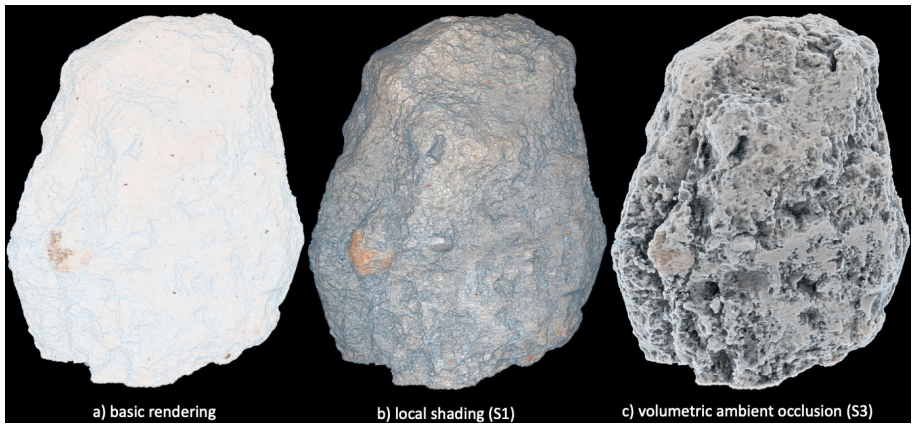


Figure 2: Soil sample renditions using IndeX XAC operators. (a) The original volume with basic rendering and no shading. (b) The volume rendered with XAC local shading (S1). (c) The more advanced XAC shading scheme with ambient occlusion (S3). Data courtesy: MOGNO beamline/Sirius, LNLS/CNPEM.

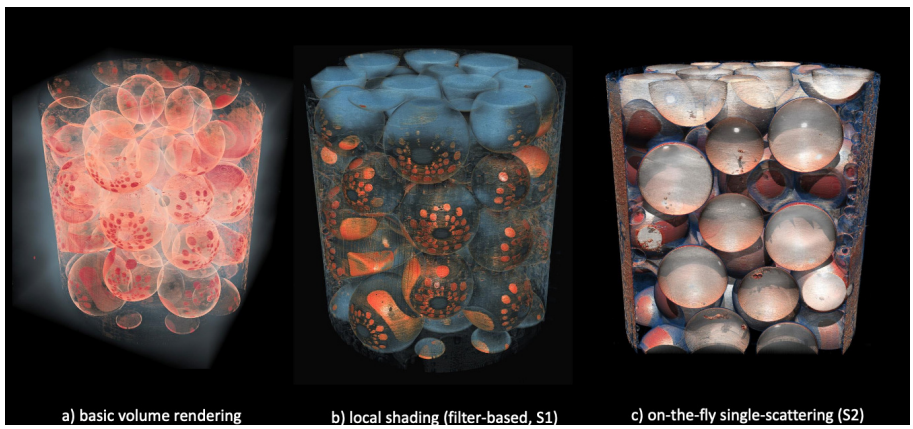


Figure 3: Silica bead fluid flow experiment renditions using IndeX XAC operators. The original image with segmentation of the gas phase is depicted in Fig. 4 (left). Data courtesy: MOGNO beamline/Sirius, LNLS/CNPEM.

TensorRT-optimized pre-trained CNNs into IndeX, such that our models provide on-the-fly segmentation of the data that is used to enhanced visualization.

RemoteVis is very well suited for use in synchrotron light sources, although it is not limited to those use cases. RemoteVis and IndeX can be used in HPC environments where large data sets are produced and require fast and easy to use tools for visualization.

ACKNOWLEDGEMENTS

We would like to acknowledge the Brazilian Ministry of Science, Technology, and Innovation (MCTI) for funding this work, through the Brazilian Center for Research in Energy and Materials (CNPEM). We would also like to thank the MOGNO and EMA beamline groups from Sirius/LNLS-CNPEM for the data used in this paper.

REFERENCES

[1] *NVIDIA IndeX*, Sep. 2021. https://developer.nvidia.com/nvidia-index

[2] *Neuroglancer*, Sep. 2021. https://github.com/google/neuroglancer

[3] N. Sofroniew *et al.*, *Napari/napari: 0.4.11rc3*, version v0.4.11rc3, Sep. 2021. DOI: 10.5281/zenodo.5349926. https://doi.org/10.5281/zenodo.5349926

[4] B. Schmid, J. Schindelin, A. Cardona, M. Longair, and M. Heisenberg, "A high-level 3d visualization api for java and imagej," *BMC Bioinformatics*, vol. 11, no. 1, p. 274, 2010. DOI: 10.1186/1471-2105-11-274. https://doi.org/10.1186/1471-2105-11-274

[5] A. Fedorov *et al.*, "3d slicer as an image computing platform for the quantitative imaging network.," eng, *Magn. Reson. Imaging*, vol. 30, no. 9, pp. 1323–1341, Nov. 2012, ISSN: 1873-5894 (Electronic); 0730-725X (Print); 0730-725X (Linking). DOI: 10.1016/j.mri.2012.05.001.

[6] *Dragonfly*, Sep. 2021. https://www.theobjects.com/index.html

[7] *OpenInventor*, Sep. 2021. https://www.openinventor.com

[8] E. X. Miqueles, G. Martinez Jr., and P. P. Guerrero, "Fast image reconstruction at a synchrotron laboratory," in *SIAM Conf. Parallel Proc. Scientific Comp.* 2020, pp. 24–34. DOI: 10.1137/1.9781611976137.3. https://epubs.siam.org/doi/pdf/10.1137/1.9781611976137.3. https://epubs.siam.org/doi/abs/10.1137/1.9781611976137.3

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBL05

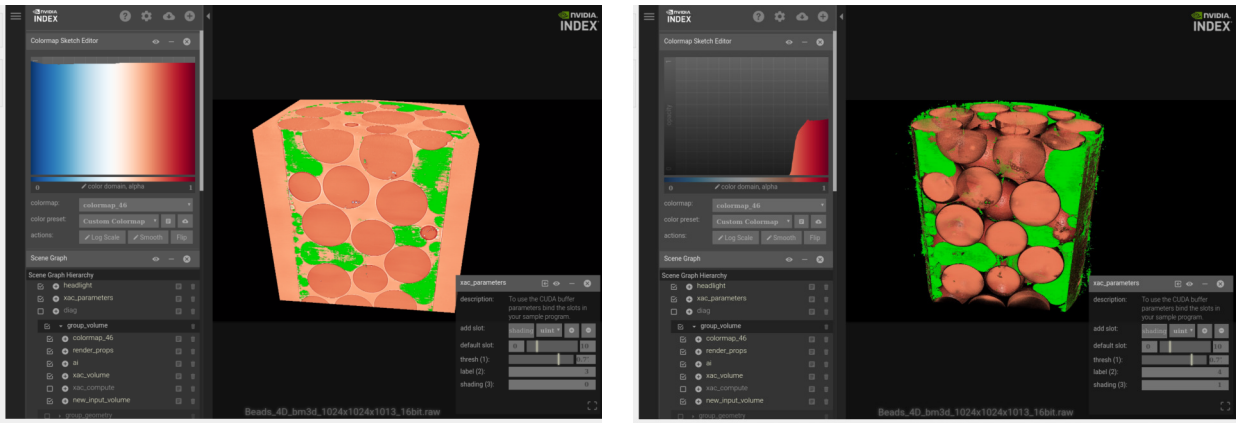


Figure 4: Integration between a pre-trained U-net segmentation model, optimized with NVIDIA TensorRT, and IndeX for visualization. The TensorRT inference engine is called to segment the gas phase of the sample (left) and the result is immediately used in the 3D rendering (right). Data courtesy: MOGNO beamline/Sirius, LNLS/CNPEM.

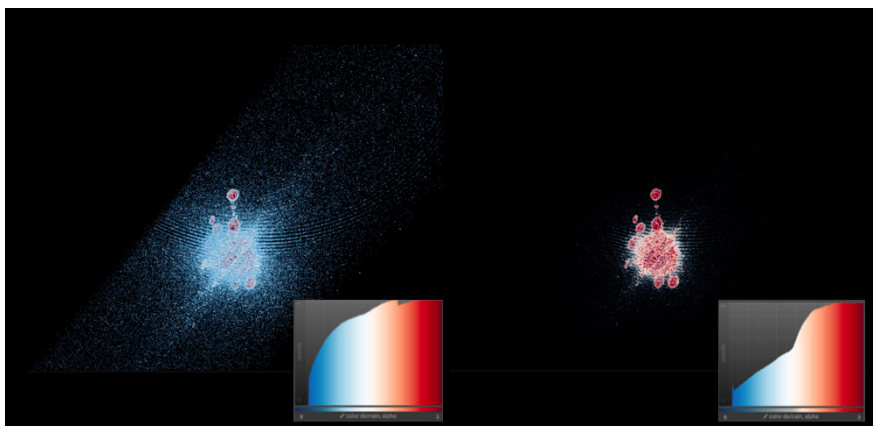


Figure 5: Point cloud renditions by NVIDIA IndeX as a particle volume. The `XYZ` and point value data are transferred by RemoteVis using SSC-RDMA and the points are rendered as spheres with radii proportional to their values. The user can then select which points to view based on their radii (left) by simply altering the considered colormap (right). Data courtesy: EMA beamline/Sirius, LNLS/CNPEM.

[9] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. Clifton Park, NY, USA: Kitware, Inc., 2015, ISBN: 1930934300.

[10] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60, ISBN: 978-3-540-39727-4.

[11] T. Bedeir, *Basic flow control for rdma transfers*, Accessed on Sep 4, 2021, Jan. 2013. `http://www.hpcadvisorycouncil.com/pdf/vendor_content/basic-flow-control-for-rdma-transfers.pdf`

[12] G. Miller, "Efficient algorithms for local and global accessibility shading," in *SIGGRAPH*, ser. SIGGRAPH '94, New York, NY, USA: ACM, 1994, pp. 319–326, ISBN: 0897916670. DOI: `10.1145/192161.192244`. `https://doi.org/10.1145/192161.192244`

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR01

PROCESS AUTOMATION AT SOLEIL: TWO APPLICATIONS USING ROBOT MANIPULATORS

L.E. Munoz*, Y-M. Abiven, F. Briquez, J. Da Silva, E. Elkaim, A. Noureddine,
V. Pinty, M. Valléau, Synchrotron SOLEIL, Saint-Aubin, France
S. Bouvel, EFOR, Levallois Perret, France

Abstract

Robot manipulators are an important component in most autonomous systems in the industry. Arc welding, machine tending, painting, picking, are only some examples where the robot manipulators are widely employed. In Synchrotrons some process can benefit from robotic approaches in order to improve automation. Automatic Sample Changer on beamlines is the most common example of automation. This paper describes two robotic applications developed at Synchrotron SOLEIL. Both applications use the SOLEIL robotic standard introduced some years ago [1]. The first application aims to automate the exchange of samples for powder diffraction experiment on the CRISTAL beamline. Hence, a pick-and-place robot is used to automate the process of picking up the sample holders and placing them on the goniometer. The second application, also of the pick-and-place type, is dedicated to the automation of the magnetic characterization of magnet modules of an U15 undulator. These modules, built with a permanent magnet and two poles, are measured using a pulsed wire method [2]. In this case, the robot picks the modules stored in boxes to then place them on the test bench of the U15 undulator.

INTRODUCTION

According to the International Federation of Robotics (IFR) an industrial robot is an automatically controlled, reprogrammable multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications (ISO 8373:2012). At the beginning, since manipulators had no external sensing, they were used for simple tasks as pick and place, mainly doing monotonous, repetitive and dangerous tasks for humans. As a result of technological advances, robots could handle more complex motion and had external sensors and then, more complex applications followed like welding, painting, grinding and assembly. Nowadays, the use of an industrial robot, along with Computer-Aided Design (CAD) systems and Computer-Aided Manufacturing (CAM) systems, not only characterize the latest trends in process automation in the industry [3], but the robot manipulators are becoming essential components in various growing sectors such as medical.

In a synchrotron facility, the most common application of industrial robots is to use the manipulator as a sample changer. The principle of a robotic sample changer is to take samples from one place and put them in another place with accuracy and repeatability. Thus, these robotic exchangers are widely used in experimental stations for Macromolecular Crystallography (MX), like is the case of the beamlines AMX and FMX at the National Synchrotron Light Source II (NSLS-II) [4], the beamlines I03, I04, I04-1 and I24 at Diamond Light Source (DSL) [5], the beamlines ID23-1 and ID23-2 at the European Synchrotron Radiation Facility (ESRF), among other MX beamlines. Some other techniques like biological Small-Angle X-ray Scattering (bioSAXS) at NSLS-II [4] and the powder X-Ray diffraction at the ESRF [6] integrate the sample changers to sample automation.

Nevertheless, the use of industrial robots is not limited to the sample changer, they can also be used to detector positioning: for Bragg CDI and Bragg-ptychography [7], to study structural dynamics with X-ray techniques [8], to enable coherent diffraction and SAXS experiments [9]; or even, robot manipulators can execute similar tasks to those present in the industry, such high precision manufacturing [10].

At Synchrotron SOLEIL, two robotic sample changers were installed on the beamlines PROXIMA-1 [11], and PROXIMA-2 [12], long before SOLEIL developed the robot standardization in 2019. This standardization, see Fig. 1, designed as part of a larger strategy in process automation, defines a robotic standard on both hardware and software which is versatile enough to cover the synchrotron requirements, while being easy to implement, to employ and to maintain in operation [1].

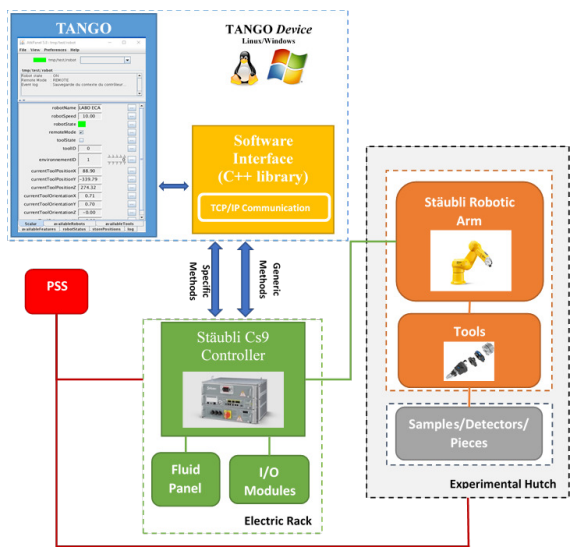


Figure 1: SOLEIL Robot Standardization Scheme.

* laura.munoz-hernandez@synchrotron-soleil.fr

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR01

The robotic standard has allowed a rapid development and deployment of two new applications at SOLEIL. The first one, integrated by IRELEC company, is dedicated to accomplishing powder X-Ray diffraction measurements, where the robot is in charge of automating the sample changer. The second one, developed entirely in-house, differs from the application examples given above, since in this application the robot manipulator is used to automate the magnetic characterization of the modules of an U15 Undulator. Since measuring the magnetic field each of the modules is time-consuming and is often a repetitive and tedious task for human operators, the automation of this task can be perfectly achieved by an industrial robot.

In the following sections these two robotic applications are described in detail.

CRISTAL BEAMLINE ROBOTIC APPLICATION

CRISTAL is an Undulator-based X-ray multi-technique diffraction beamline, which delivers high-brightness beam in the 5 to 30 keV energy range. It is dedicated to study single crystals and powders. All the standard techniques for structural analysis on single crystals and powders are implemented, as well as more advanced techniques like coherent diffraction imaging and time-resolved diffraction in a pump-probe scheme [13].

To improve the efficiency of powder diffraction measurements, the process has been automated using an industrial robot:

1. Automatic mounting of capillaries. The robot is in charge of mounting the capillaries to be studied on a 2-circle diffractometer, following the workflow shown in Fig. 2.

2. Sample self-centering task. The diffractometer is equipped with a spinner and a Y-Z translation stage. Both of them serve to the sample self-centering[1].

3. Diffraction measurements. Measurements are performed by scanning the 2-circle diffractometer. The diffractometer includes two detectors. In order to obtain high-angular-resolution diagrams in less than one hour for ab-initio structure determinations a multi-crystal analyzer detector (21-Si(111) crystals) is employed. For phase analysis, in-situ or operando studies, a Mythen (from Dectris) curved pixel detector is used.

4. Automatic dismounting of capillaries. The robot returns the samples to their place in the sample store.

This process can be repeated without entering the experimental hutch as many times as necessary until the maximum capacity of the sample store is reached.

[1] How self-centering is implemented is outside the scope of this paper.

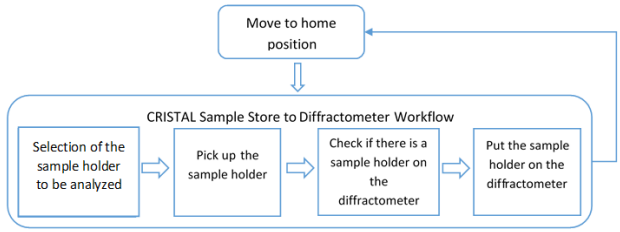


Figure 2: Robot Workflow: Sample Store to Diffractometer.

Robotic Sample Changer

The robotic platform, shown in Fig. 3, is based on a Stäubli TX2-60L robot with six degrees of freedom, which is mounted on a mobile chassis and integrates the Cs9 robot controller, the electric rack, the fluid panel and the input/output modules. The TX2-60L was chosen for its radius of work of 920 mm and the load capacity of 3.7 Kg. sufficient enough to carry the robot end-effectors. The main end-effector is composed of a pneumatic tool changing system, a 3-finger centric gripper, a collision and an overload protection system and a laser sensor. The manipulator is equipped with solenoid valves that operate the gripper, the tool changing system and the collision device without external devices. The robot controller is interlocked with the beamline Personnel Safety System (PSS) such that no robotic movements are allowed unless the hutch has been locked.

The sample store is placed next to the robot base and it has a capacity of 36 samples. The samples are mounted on 17mm diameter cylindrical sample holders. It can be observed from Fig. 4 that the lower part of the sample holder has a female cone that is used to place it on the sample store. The TX2-60L robot has a positioning repeatability of 30 μm and combined with a magnet housed in the bottom of the female cone, allows the accurate location of the sample holder on the diffractometer.

Although the position of the diffractometer is known, it has 1 degree of freedom (translation), thus the robot must be able to detect if it has been displaced. Furthermore, since the robot can be itself moved, a calibration process had to be implemented.

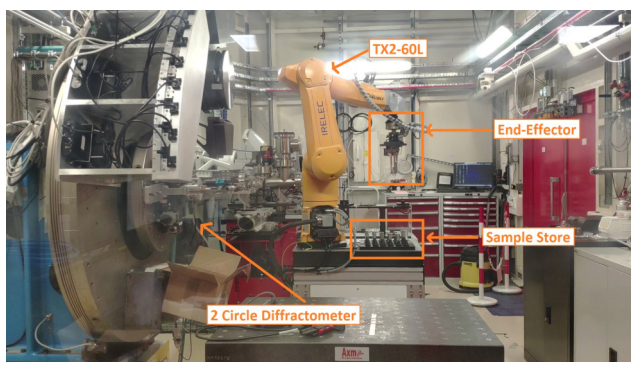


Figure 3: CRISTAL Robotic Sample Changer.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

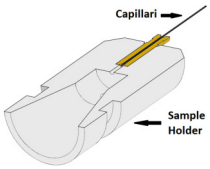


Figure 4: CRISTAL Sample Holder.

Calibration consists of measuring, with the help of a robot's tool equipped with a laser sensor (Keyence IL-30), the center of the placement area of the sample holder, located in the center of the diffractometer. If there is any difference between the calculated value and the value saved in the memory of the controller, the value save in memory is updated.

As it is described below in the **SOFTWARE INTEGRATION** Section, the robotic standard guarantees some flexibility in integration. Therefore, the calibration of the position of the diffractometer can be carried out when some movement in the translation of the diffractometer is performed or systematically at the beginning of a measurement campaign, i.e. before the execution of the 4 steps introduced above.

It is noteworthy that the execution time of the two first steps is about 1 minute and 30 seconds. This allows saving 5 minutes to 10 minutes if a human operator performs these tasks. In addition, the automated process gives the opportunity to almost uninterrupted use the beam 24 hours a day during the experiments.

MAGNETIC CHARACTERIZATION ROBOTIC APPLICATION

In order to achieve high-performance Undulators, a magnetic characterization of the modules (a hybrid magnetic structure, made out of permanent magnets and iron poles) that compose the insertion device have to be accomplished. For each module, the magnetic field is measured with a Hall effect probe mounted on a trolley which moves on a rail, and a rotating coil, supported by a set of motorized stages, is responsible of the magnetic field integral measurements. These two measuring instruments are placed next to the magnetic measurement bench, where each module is positioned on the internal girder for the purpose of being measured. Despite the fact that the magnetic field measurements are done automatically, the positioning of the module on the magnetic measurement bench was effectuated until now, by a human operator. Performing this task is time-consuming and it becomes repetitive for the operator due to the hundreds of modules that have to be characterized. Hence, to fully automate the magnetic characterization, a robotic solution to position the modules was implemented. The simplify workflow of the automated process is shown in Fig. 5. Where the orange dotted-line blocks are the tasks performed by the robot and the blue dashed-line blocks by the other devices of the test bench.

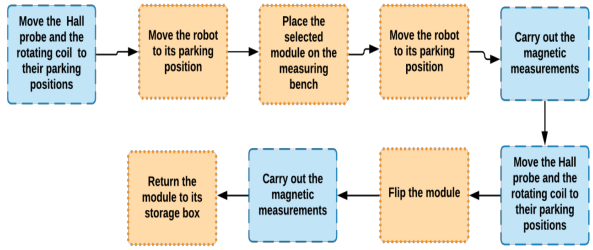


Figure 5: Automatic Magnetic Measurements Workflow.

Robotic Platform

The robotic platform is composed of a robot Stäubli TX2-60L mounted on a pedestal. The end-effector is formed by an electric 2-finger parallel gripper, a manual changing system and an ATI Force/Torque sensor system. The electric rack, the Cs9 robot controller and the Input/Output modules are placed in a separate bay. An optical table located next to the robot supports four boxes where 120 modules (30 per box) are arranged vertically. This optical table also supports a drop-off area that had to be implemented to be able to place the modules correctly on the measurement bench, see Fig. 6.

Since there are magnetic constraints, both the mechanical design and the design of the robot trajectories, were conceived taking into account these constraints. In addition, the gripper jaws were designed to be able to grasp the modules in two different ways, as it can be observed in Fig. 7. The "first configuration", Fig. 7a, takes place when the modules are taken out of the boxes and positioned on the drop-off area; while the "second configuration", Fig. 7b, is used when the robot takes the modules from the drop-off area and places them on the measurement bench and vice versa.

The Force/Torque sensor is used to ensure the accurate positioning of the module on the measurement bench. Once the module is released from the gripper, a series of movements is performed by the robot in such a way that the jaws push the module in the $X-Y$ direction until a $X-Y$ force threshold is reached.

The characterization of a module takes around 15 min, therefore the robot-based bench can work up to 30 hours without the need for human intervention.

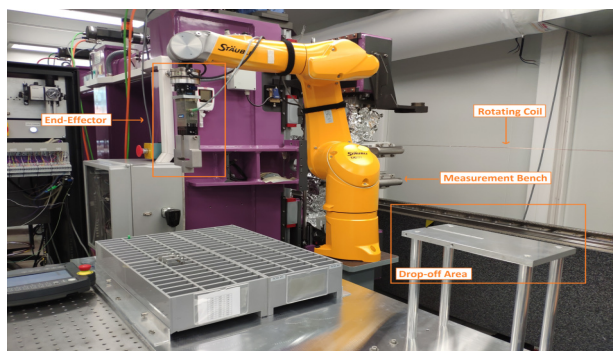
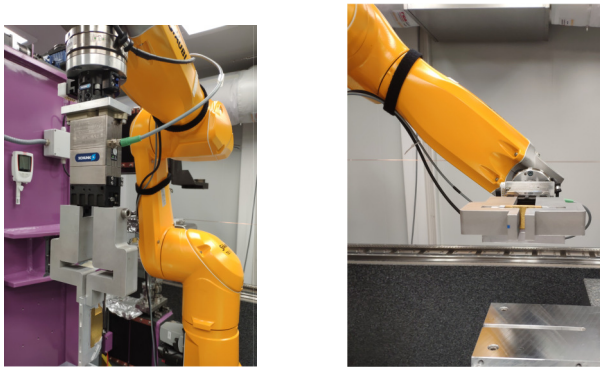


Figure 6: Robotic Platform.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR01



(a) Gripper Configuration 1. (b) Gripper Configuration 2.

Figure 7: Two-in-one Gripper.

SOFTWARE INTEGRATION

It can be observed in Fig. 1 that a C++ library interface was developed to enable communication between TANGO and the robot controller. The library integrated into the TANGO Device communicates with the Cs9 controller via the Ethernet network using TCP/IP.

In order to facilitate the integration into the control system, a series of methods or routines are used to communicate and act on the physical environment of the robot. The so-called generic methods are instructions that can be applied to any robotic application, such as, turn the robot on/off, sending the robot to a parking position or home position. On the other hand, the specific methods are instructions dedicated to an application, i.e., they execute functionalities or features developed for a single application.

For the CRISTAL robotic application some of the implemented features are: *put* (take the sample from the sample store at coordinates *row,column*, and mount it on the diffractometer), *get* (take the sample from the diffractometer and put it back at its position in the sample store), *teachgonio* (launch the automatic teaching of the diffractometer position, i.e. the calibration process). For the magnetic characterization some of the implemented features are: *put* (take the module from the selected box at coordinates *box,row,column*), *get* (take the module from the measurement bench, flip it, and put it back at its position), *flip* (take the module from the measurement bench, flip it, and put it back on the measurement bench).

TANGO allows to individually manage each component of the experimental setups, i.e. the translation of the diffractometer, the spinner, and so on, in the case of the CRISTAL application; the Hall probe and the rotating coil in the case of the magnetic measurements. Thus, the whole process is manage at a higher level using the different TANGO bindings to other languages, see Fig. 8. This approach then provides that the process can be adapted and can evolve according to the needs, without having to modify the code programmed in the robot controller.

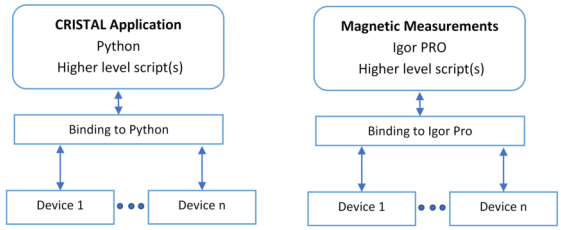


Figure 8: Software Integration Schematic.

CONCLUSIONS

The standardization of robots has made possible to develop robotic applications in a relatively short period of time, considering that one of them was completely designed and developed in-house. These two applications have established the technical basis for the future development of robotic applications. Ten other applications have already been identified to include industrial robots in the automation of experiments. These identified applications are not only of the pick-and-place type, but also include the positioning of detectors and new assembly methods for SOLEIL future upgrade, which means that new challenges must be faced in the short and long term, notably avoiding obstacles. Besides, the sample environments are increasingly complex and more demanding in performance, reliability and safety. The integration of robot manipulators in the sample environments therefore becomes another challenge.

In order to better address the aforementioned challenges, SOLEIL is already working on some Collision Avoidance Systems (CAS) approaches, especially for systems who involves in limited workspaces, such as the sample environments. In the near future, adding CAS to the industrial robots will be an important step to accomplish.

AKNOWLEDGEMENTS

The authors thank all SOLEIL staff who have participated in one way or another in the development of the robotic applications.

REFERENCES

[1] Y.-M. Abiven *et al.*, "Robotizing SOLEIL Beamlines to Improve Experiments Automation", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 183–186. doi:10.18429/JACoW-ICALEPCS2019-MOPHA001.

[2] M.Valléau *et al.*, "Measurements of soleil insertion devices using pulsed wire method", in *2nd International Particle Accelerator Conference*, San Sebastián, Spain, 2011, pp. 3242–3244.

[3] J. J. Craig, "Introduction to Robotics: Mechanics and Control", Pearson, ISBN 013-123-629-6, 2005.

[4] E.O. Lazo *et al.*, "Robotic sample changers for macromolecular X-ray crystallography and biological small-angle X-ray scattering at the National Synchrotron Light Source II", *J. Synchrotron Rad.*, vol. 28, pp. 1649-1661, 2021.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

[5] BART Diamond, https://www.diamond.ac.uk/industry/Industry-News/Latest-News/Synchrotron-Industry-News---MXnews5/BART-update.html

[6] ID22 Beamline, https://www.esrf.fr/id22

[7] C. Rau *et al.*, "New Imaging Opportunities at the DIAMOND Beamline I13L", in *Proc. SPIE 11112, X-Ray Nanoimaging: Instruments and Methods IV,11120L*, Oct. 2019. doi:10.1117/12.2543799.

[8] M. Chollet *et al.*, "The X-ray Pump–Probe instrument at the Linac Coherent Light Source", *J. Synchrotron Rad.*, vol. 22, pp. 503–507, 2015.

[9] G. Ingold *et al.*, "SwissFEL instrument ESB femtosecond pump-probe diffraction and scattering", in *AIP Conference Proceedings*, vol. 1741, no. 1, p. 030039, 2016. doi:10.1063/1.4952862.

[10] SwissFEL, https://www.psi.ch/en/media/our-research/first-swissfel-accelerator-structure-completed

[11] PROXIMA-1 Beamline, https://www.synchrotron-soleil.fr/en/beamlines/proxima-1

[12] PROXIMA-2 Beamline, https://www.synchrotron-soleil.fr/en/beamlines/proxima-2a

[13] CRISTAL Beamline, https://www.synchrotron-soleil.fr/en/beamlines/cristal/

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR02

AN INTEGRATED DATA PROCESSING AND MANAGEMENT PLATFORM FOR X-RAY LIGHT SOURCE OPERATIONS *

N.M. Cook†, E. Carlin, P. Moeller, R. Nagler, B. Nash, RadiaSoft LLC Boulder, CO 80301, USA
A. Barbour, M. Rakitin, L. Wiegart, National Synchrotron Light Source II,
Brookhaven National Laboratory, NY, 11973, USA

Abstract

The design, execution, and analysis of light source experiments requires the use of increasingly complex simulation, controls and data management tools. Existing workflows require significant specialization to account for beamline-specific operations and pre-processing steps in order to collect and prepare data for more sophisticated analysis. Recent efforts to address these needs at the National Synchrotron Light Source II (NSLS-II) have resulted in the creation of the Bluesky data collection framework, an open-source library providing for experimental control and scientific data collection via high level abstraction of experimental procedures, instrument readouts, and data analysis. We present a prototype data management interface that couples with Bluesky to support guided simulation, measurement, and rapid processing operations. Initial demonstrations illustrate application to coherent X-ray scattering beamlines at the NSLS-II. We then discuss extensions of this interface to permit analysis operations across distributed computing resources, including the use of the Sirepo scientific framework, as well as Jupyter notebooks running on remote computing clusters.

INTRODUCTION

X-ray light sources are prominent drivers of scientific discovery across a range of disciplines. These facilities serve a diverse user community, often providing concurrent beam time and user support to tens of domain scientists with unique backgrounds. Increasing demand for beam time, coupled with the increasing sophistication of experiments, places constraints on the infrastructure required to successfully carry out experiments within time and resource constraints. Recently, significant development efforts have been made towards improving experimental planning and execution; however, significant challenges remain to integrating real-time analysis tools within the experimental workflow. In this proceedings, we discuss a strategy for incorporating analysis pipelines within common experimental workflows, focusing on applications at the NSLS-II light source. We present a schematic workflow for orchestrating analysis in concert with experimental execution. We then demonstrate this workflow via an open source, browser-based interface furnishing beamline agnostic analysis pipelines.

* This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC00215553.
† ncook@radiasoft.net

AN INTEGRATED FRAMEWORK FOR EXPERIMENT AND ANALYSIS

Our proposed strategy is to integrate a flexible analysis platform with a mature controls framework, leveraging pre-existing workflows and data schemes wherever possible. To this end, we have adopted the Bluesky Data Collection Framework, which is in active use across many beamlines at the NSLS-II [1]. BlueSky aims to provide end-to-end experimental planning, execution, and data acquisition tools through a set of interoperable Python libraries. We highlight a few of the critical libraries for the application discussed below. First, the eponymous `bluesky` library implements a run engine and event model to permit experimental control and data collection through the execution of high level plans. The `ophyd` library provides hardware abstraction to communicate plans to devices along the beamline. The `databroker` library implements an API for structured access to experimental data and metadata generated during an experiment executed by Bluesky.

For the analysis component, we chose to use the Sirepo platform to orchestrate execution of analysis pipelines. Sirepo is an open-source scientific computing gateway that provides access to community codes through custom, browser-based interfaces and an embedded JupyterHub instance. Sirepo is designed to be hardware agnostic; simulation environments are deployed via Docker containers, and can be executed across a range of computing systems, ranging from a laptop to a GPU cluster an high performance computing facility. Sirepo provides support for numerous accelerator modeling and related tracking codes; existing applications have been employed to provide customized simulations of X-ray beamlines at the NSLS-II using the Synchrotron Radiation Workshop code [2]. Sirepo has also been integrated with Bluesky to enable the asynchronous execution of long-running SRW simulations to support multi-parametric optimizations of beamlines [3].

Our approach is to provide support for analysis pipelines that complements Bluesky's support for experimental execution. Figure 1 depicts a relational diagram between the different components of the envisioned platform. The Sirepo API and user interface will support the design, templating, and execution of analysis software, to be run in tandem with experimental execution. Sirepo templates simulations via JSON schemas, providing descriptive metadata, as well as mechanisms for sharing simulations or downloading and reproducing them elsewhere. This approach is akin to Bluesky's event model for describing documents generated by experimental plans. Sirepo enables hardware-independent descriptions

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

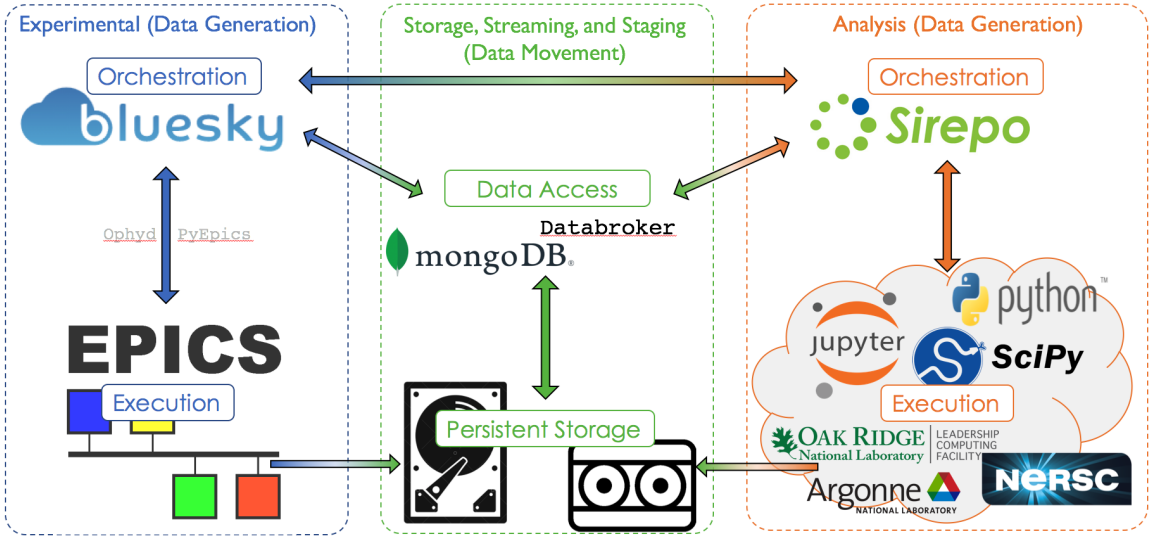


Figure 1: A high level schematic reflecting parallel implementations of Bluesky for experimental orchestration and Sirepo for analysis orchestration. While Bluesky provides schemas to define an experimental plan, along with high-level abstraction of the experimental hardware, Sirepo will provide schemas for analysis pipelines, along with access to requested computing resources, ranging from local endstations to remote high performance computing facilities. Both tools will leverage the `databroker` library to handle data selection, staging, and storage, relying on document schemas and searchable dictionaries to streamline access.

of simulations. Analyses can be executed across varied computational resources, ranging from local machines to high performance computing facilities, via the deployment of Docker images. Again, this approach has a parallel in Bluesky, which enables high-level abstraction of hardware, with low-level details handled by device-specific representations in the `Ophyd` library.

A COMMON ANALYSIS WORKFLOW

To design an analysis platform that is beamline agnostic, we have extrapolated a high-level analysis workflow predicated on typical use cases at the Coherent Soft X-Ray (CSX) and Coherent Hard X-Ray (CHX) beamlines at NSLS-II. While this workflow does not uniquely specify all possible configurations, it aims to span a representative sample of workflows, and therefore serve as a rubric for guiding interface feature design. This workflow distinguishes five steps in the analysis procedure: staging, pre-processing, analysis, documentation, and post-processing. Figure 2 describes the resulting pipeline.

Our initial prototype aims to address the first three steps, without compromising existing documentation and post-processing capabilities in place for user's at NSLS-II. Specific emphasis has been placed on accommodating the analysis step, because it presents unique demands on software flexibility and resource management. At CHX and CSX, many users leverage Jupyter notebooks running IPython kernels [4] are customized to provide near real-time analysis for a subset of the data generated during runs. Executing these notebooks requires dedicated support for required dependencies, as well as sufficient computing resources to run notebooks for each of the hundreds to thousands of datasets generated during a given experiment. Notebook execution may take 1 – 10 minutes depending upon the analysis, while data capture may only require a few seconds, meaning that the analysis step constitutes a bottleneck in completing the pipeline in real-time.

A PROTOTYPE SIREPO INTERFACE FOR INTEGRATED ANALYSIS

With this pipeline in mind, we demonstrate a prototype browser-based interface for carrying out beamline agnostic analysis at X-Ray light sources, hosted by Sirepo, and to be used in conjunction with experimental execution carried out via Bluesky. The initial prototype consists of three tabs , addressing the first three stages of the previously discussed analysis pipeline.

Figure 3 depicts the data selection tab, which permits users to browse data generated by Bluesky runs at a particular beamline. By leveraging the `databroker` library's catalog structures, runs can be searched and filtered according to primary metadata such as start and stop time, unique identifier (UID), user, and other relevant flags. The search results can be subsequently sorted, and users may select a subset of remaining runs for further processing.

Following this selection, users are guided to the metadata tab, which enables additional inspection of high level metadata regarding the run, alongside specific device and analysis details provided by the Bluesky run documents. Currently, a fixed subset of metadata is presented to the viewer. However, future implementations will expand the ability to search metadata entries using queries permitted by `databroker`'s

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR02

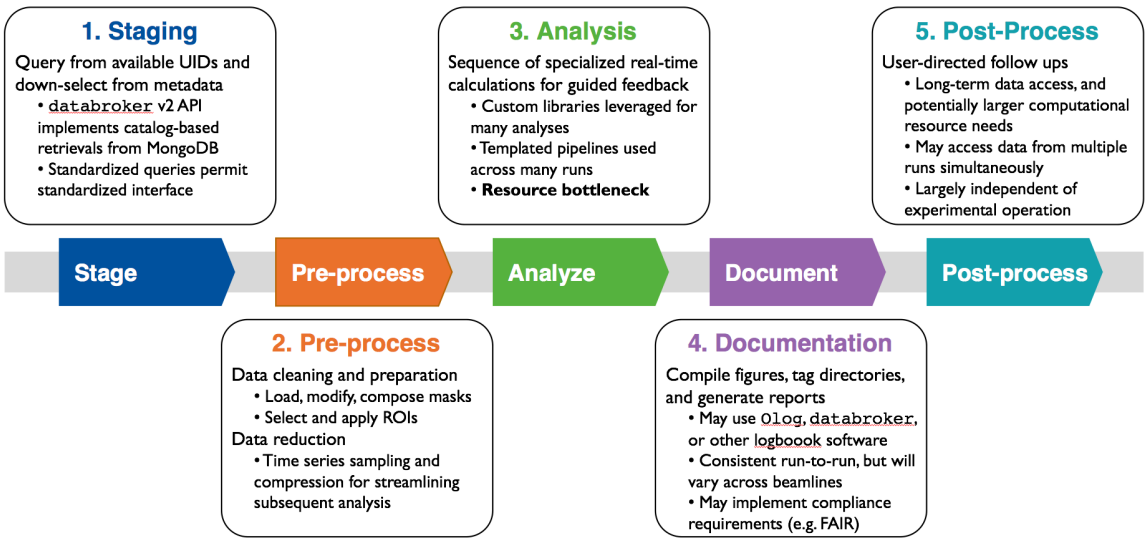


Figure 2: A high-level workflow characterizing an analysis pipeline which meets the needs for beamline experiments. During steps (1) and (2), Experimental schemas, in this case provided via `databroker`. are leveraged to provide metadata to support downselection and preparation of data for analysis. During step (3), an external codebase (e.g. Jupyter) may be employed to carry out specialized calculations. For experiments performed using Bluesky, documentation can be handled via `databroker` or `Olog`, while post-processing steps are often carried out off-site.

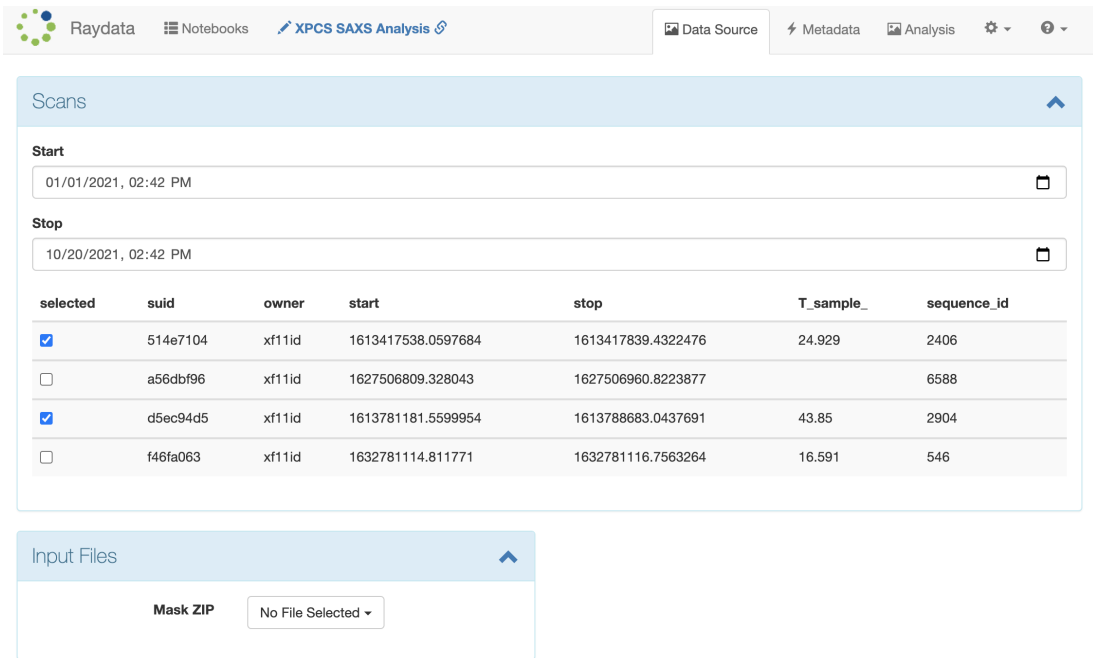


Figure 3: The data selection tab permits users to search, sort, and select from available runs in a prescribed catalog, using high level metadata to guide decisions about selections.

mongoDB-based implementation. Figure 4 illustrates the use of this interface for inspecting two selected datasets. Further improvements will permit quick snapshots of raw data files as guided by the user.

Lastly, the analysis tab addresses the analysis portion of the pipeline. For a given dataset, a user-specified Jupyter notebook is executed to carry out any specialized analysis of the initial datasets. The notebooks are executed via Docker images designed to include all relevant dependencies for the calculations, permitting their deployment across distributed computational resources. Products of the analysis notebook, such as figures, are inspected by the interface and presented to the user immediately, enabling real-time feedback during execution. Figure 5 depicts a snapshot of the analysis tab, illustrating the generation of figure panels reflecting the products of the notebook.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR02

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

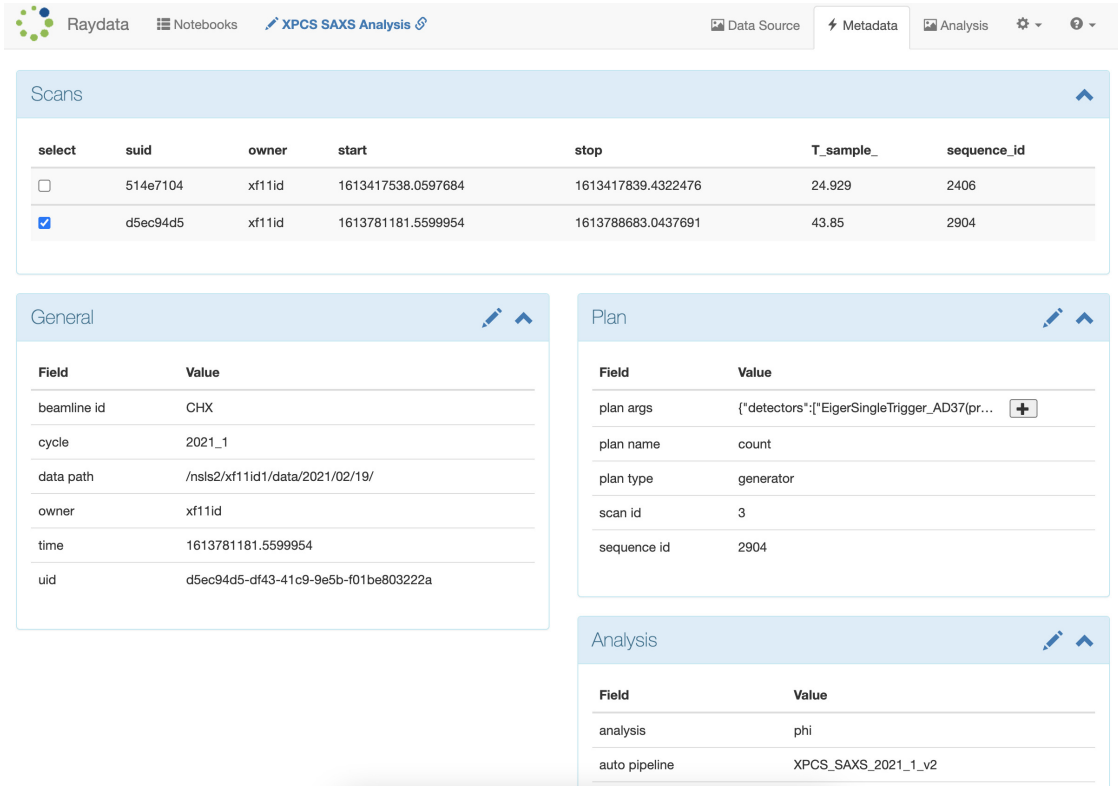


Figure 4: The metadata inspection tab provides additional details from the run metadata and start documents to permit inspection of selected runs.

Figure 5: The analysis tab of the interface, displaying previews of several of the figures generated by running the analysis notebook for this particular run.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR02

CONCLUSION

We report on progress in the development of an interactive user interface for real-time analysis of X-ray light source experiments, built atop the Bluesky framework and Sirepo platforms. The interface leverages shared metadata structures provided by the Bluesky, making use of the `databroker` library to enable active searching, sorting, and retrieval of runs from catalog structures generated during experimental execution. Custom analysis routines are supported by deploying templated Jupyter notebooks via Docker images containing the necessary dependencies. Job management, resource allocation, and queueing are provided by Sirepo, along with real-time feedback through automated figure previews and reports. Future developments will enhance queueing capabilities to support user-directed screening and prioritization of data, with the goal of providing an integrated, automated, and customizable analysis workflow to complement Blueksy's experimental procedures on active beamlines at NSLS-II and elsewhere.

REFERENCES

[1] Bluesky Project, `https://blueskyproject.io`

[2] Rakitin M S *et al.*, "Sirepo: an open-source cloud-based software interface for X-ray source and optics simulations", *J. Synchrotron Radiat.*, vol. 25, pp. 1877–1892, 2018.

[3] Rakitin M S *et al.*, "Introduction of the Sirepo-Bluesky interface and its application to the optimization problems", *Proc. SPIE*, vol. 11493, 2020. `doi:10.1117/12.2569000`

[4] Pérez, F. and Granger B., "IPython: A System for Interactive Scientific Computing", *Comput Sci. Eng.*, vol. 9 pp. 21–29, 2007.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR03

STATUS OF BLUESKY DEPLOYMENT AT BESSY II*

William Smith†, Sebastian Kazarski, Roland Müller, Luis Vera Ramirez, Pierre Schnizer, Simone Vadilonga
(HZB, Berlin)
Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Berlin, Germany

Abstract

The modernization plan for the experimental DAQ at the BESSY II [1] is underpinned by the capabilities provided by the Bluesky software ecosystem [2]. To interface with the hardware Bluesky relies on the Ophyd library, that provides a consistent high-level interface across a wide-range of devices. Many elements of the accelerator, some beamlines and endstations are adopting the Bluesky software. To meet FAIR data obligations, the capture of metadata with Bluesky and the export into a permanent and easily accessible storage called ICAT are investigated [3]. Finally, initial studies to investigate the integration of ML methods, like reinforcement learning [4] were performed.This paper reports on the work that has been done so far at BESSY II to adopt Bluesky, problems that have been overcome and lessons learned.

INTRODUCTION

The modernization strategy for experimental DAQ at BESSY II [1] uses EPICS as the unique integration layer for subsystems and software packages. There are many different solutions in use on the experimental floor for the layer above EPICS at BESSY II. This layer has to facilitate experimental flow control, data and metadata collection, storage and analysis.

While various home-grown solutions are in use at BESSY II, most beamlines use spec [5]. This popular and flexible tool has an easy to learn command line interface, has been deployed for decades in production at facilities around the world and is known and understood by many beamline staff. However it is not open source, has a tiny developer community, uses a language that is not well known outside the research community and has no error checking.

Open source alternatives including Sardana [6], PShell [7] and Bluesky [2] were all considered. The Bluesky software ecosystem was judged the most promising tool. Many other synchrotron facilities in Europe (Alba, PETRA III, ESRF, ELI NP, MAX IV, SOLARIS ...) use Sardana interfacing with TANGO. In principle it's possible to use EPICS with Sardana, but the underlying data models of the frameworks differ extremly. Due to the complex adaptation effort required, no facility is doing this in production. Bluesky and Ophyd interface natively with EPICS and there is a large user community in the US (NSLS II, APS, LCLS, SSRL, ALS) and a growing community in Europe (BESSY II, MPG/FHI, Diamond, PSI/SLS ...) and around the world (CLS, ANSTO, PLS II). Like Sardana, it's also based on Python. The data model of Ophyd devices is close to a corresponding TANGO device server instance. That might open opportunities to integrate complex TANGO units into Bluesky or take advantage of Sardana controlling Ophyd devices. Using a tool based on well known language with widely available training has made it easier to bring new people into the project.

This paper will report on progress in the deployment of Bluesky at the facility. First background on the infrastructure of BESSY II and a specific case study, the Energy Materials In situ Lab (EMIL) beamlines, is described. Then component integration, experimental flow control, data and metadata collection, user interfaces, and integration with machine learning (ML) tools are all explored.

SPECIFICS OF BESSY II

From the accelerator commissioning test bed, Bluesky made its way to the experimental floor via instrument integration at the EMIL beamlines. With the aim of being able to collect metadata about the state of the entire beamline when running Bluesky plans, each element of the beamline was given a Python device abstraction using the Ophyd Python package. These general devices classes were then easily transferred to create an interface for a novel beamline project (U49/2 PGM-2, "Aquarius") and to aid replacement efforts at a spec [5] automated beamline (μSpot). Work now continues to integrate other beamlines.

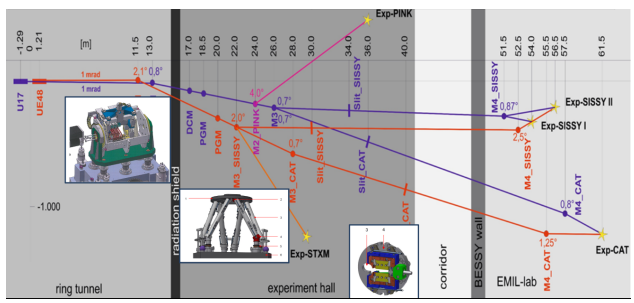


Figure 1: Complex beamline switch yard of the EMIL featuring multiple end stations, instruments and use cases.

CASE STUDY ON EMIL

The initial work to investigate deploying Bluesky at the beamlines was carried out at EMIL. EMIL comes with numerous additional challenges that are due to the complexity of the set-up and the variety of instruments to serve. Plain vanilla EPICS device control of the beamlines comes with EDM based engineering screens, the monochromator controls is based on an integrated VME application, obscur-

* Work funded by BMBF and Land Berlin
† william.smith@helmholtz-berlin.de

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR03

ing motor controls, beamline commissioning relied on spec macros, continuous energy scan requirements are not eased by custom hardware connectivity, the instruments are owned by the independent legal bodies HZB and MPG. Nevertheless the whole EMIL set-up can be seen as a pathfinder project easing the general roll-out of Bluesky to the inventory of BESSY II beamlines.

COMPONENT INTEGRATION

Ophyd allows us to solve one of the biggest problems with EPICS V3, that records have no inherent structure. EPICS V7 addresses this problem but it's not currently widely used outside areaDetector. Ophyd provides a framework to collect various signals connected to records together into groups which together form devices. It's then possible to read the status of an entire device in one go, and importantly for the collection of metadata it automatically gives context to individual parameters.

It does not solve the problem that values of components of a device can change while they are being read. This can only be solved at a lower level than the IOC to ensure that parameters update synchronously.

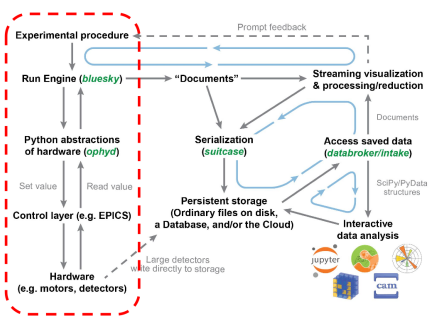


Figure 2: Device integration, abstraction.

Following the example set by NSLS II, LCLS and other labs a set of ophyd devices were created for every element of the EMIL beamlines. This was challenging because of a lack of standardization between different EPICS interfaces for different devices, and even within the same device types at BESSY. None of the motors on the beamline used the EPICS Motor Record, which meant that bespoke device classes had to be made and tested for each type. Despite this challenge the Ophyd framework proved very flexible, all devices were eventually integrated.

Of particular note were the devices on the beamline and at the end station which used the epics areaDetector interface. This included electron analyzers, cameras and spectrometers. Ophyd provided relatively easy integration for all of these, as well as providing the ability to use and control the areaDetector plugins.

Having created Python interfaces for devices, the use of pytest [8] for device level integration was investigated. The existing suite of tests used by the Ophyd package help to guarantee that the abstraction interface itself works as expected. Tests were created to test that physical systems operated correctly. This was particularly important for the positioner device classes that had to be created because the standard EpicsMotor interface was not available. Various bugs that would otherwise have been missed were found by test scripts that attempted to move every motor on the beamline. More work is needed in this area at the beamlines but the ability to perform full integration level tests from Ophyd, through EPICS and to real hardware is promising.

In the accelerator, tools for aperture scans and a power supply multiplexer were developed and tested using a digital twin with an Ophyd interface to simulated hardware. By changing the EPICS PV name prefix passed to the Ophyd device it could be switched to point at the real hardware. A similar approach was used to develop a continuous energy scan plan for the beamlines using Ophyd to connect first to simulated and then real hardware. Digital twins were critical to the development of these tools because there are not spare monochromators or accelerators to test on, and the time given for commissioning is very limited.

EXPERIMENTAL FLOW CONTROL

The commissioning of the EMIL beamlines was performed with spec. The task of converting these spec macros to Bluesky plans was relatively painless. Common spec macros like ascan have similar Bluesky equivalents. More bespoke macros that set up various elements of a beamline could also be easily converted.

Ultimately a beam time manager state machine will be needed, orchestrating sub use cases and handling the variety of experimental plans. This will be based on scripts and semaphores at the point of state transition.

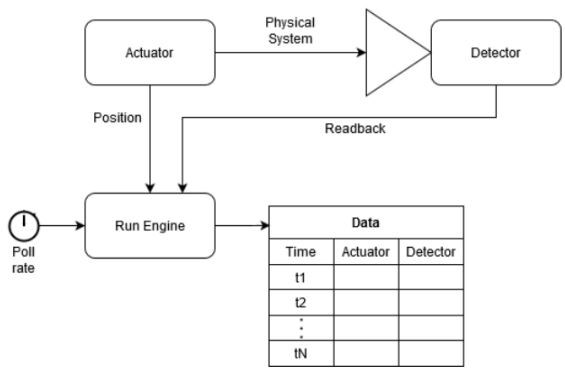


Figure 3: Values are grouped together in continuous energy scans by polling from the Bluesky run engine.

Continuous energy scans, in which a monochromator is commanded to move between certain energies at a certain rate while values are read from detectors on the beamline is implemented in a Bluesky plan. Values from the detector(s) and the energy of the monochromator are polled by the run engine at a fixed rate. The monochromator is assumed to move slowly enough that the error introduced by the energy and detectors being read at slightly different times is insignif-

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

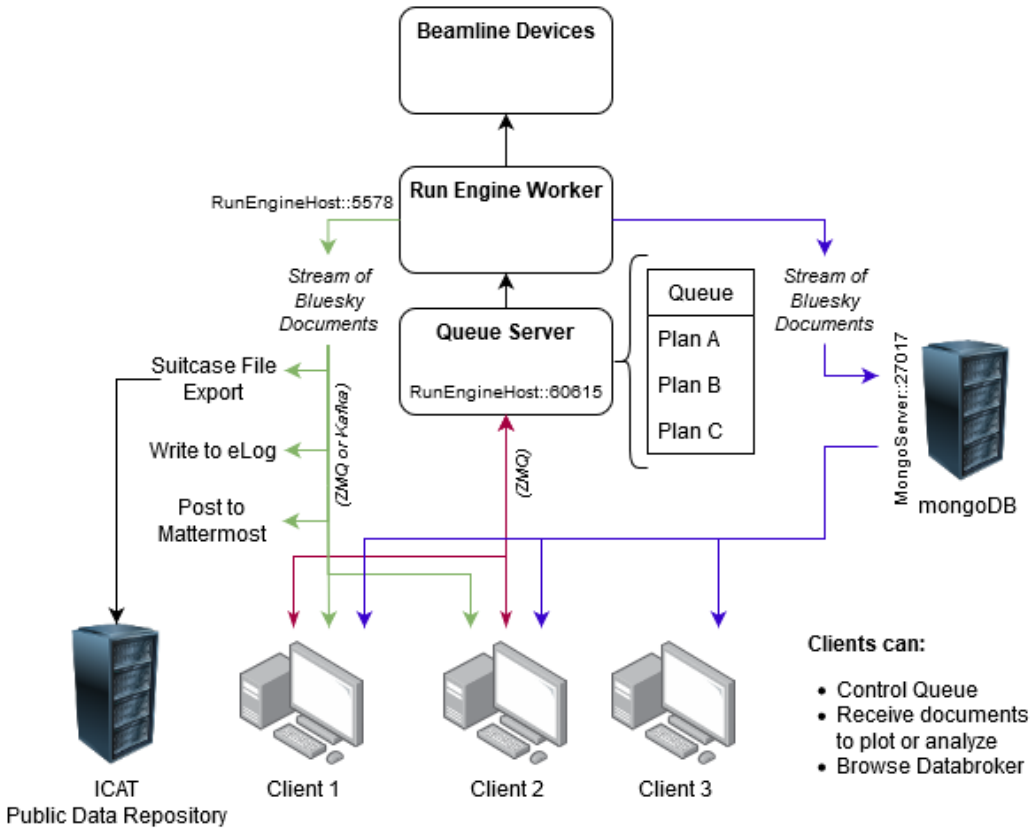


Figure 4: Connections between run engine and clients.

icant. Values of energy and detector readouts are presented live to the user. Bluesky offers a flexible framework for implementing this kind of scan, allowing for the integration of hardware trigger systems to replace polling where necessary in the future.

At both the beamlines and the accelerator the Bluesky run engine is currently interacted with directly, i.e it runs in the same process as the IPython session or Jupyter Notebook that it's run from. This is sufficient for our current use cases, but we are investigating running the run engine as a service using the Bluesky-Queueserver [9] package developed by NSLS-II (See Fig. 4). Operating the run engine as a service means that high importance tasks like data acquisition can be separated completely from less important tasks like displaying live data to clients, or writing to an eLog. If a graphical interface crashes, the run engine will keep performing the experiment.

The Queueserver additionally gives the ability for multiple clients to queue plans, and observe what is currently running. For complex beamlines like EMIL, user access rights for different beamlines or endstations change regularly between shifts. For example, in one shift one endstation might need control of both beamlines, and in the next two different endstations might need to control one beamline each. Using existing access controls based on the EPICS channel access gateway would be one solution, but is seen as being too coarse and is problematic to implement on endstation machines which tend to have common experimental user accounts. It's hoped that adding logic based on agreed scheduling to the Queueserver and user authentication of the clients will allow us to avoid problems we currently experience with multiple simultaneous remote users affecting each other. Users should be allowed to run only particular plans with particular devices during times agreed beforehand. This idea will require considerable development.

DATA AND METADATA

Documents generated by the Bluesky run engine are saved in a mongoDB. For scaler values and most waveforms, data is saved directly in the database. For images produced by detectors with an EPICS areaDetector [10] interface, Bluesky can instruct the EPICS IOC to save images on fast storage with unique names defined by the run engine. Those unique names and file locations are then saved in the metadata of the run stored in the mongoDB. When a client later wants to access the data from a run using the unique run identifier the Databroker [11] package fetches the image from the storage location and presents it to the client. To the client this is transparent, it is as if the image was saved in the mongoDB.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR03

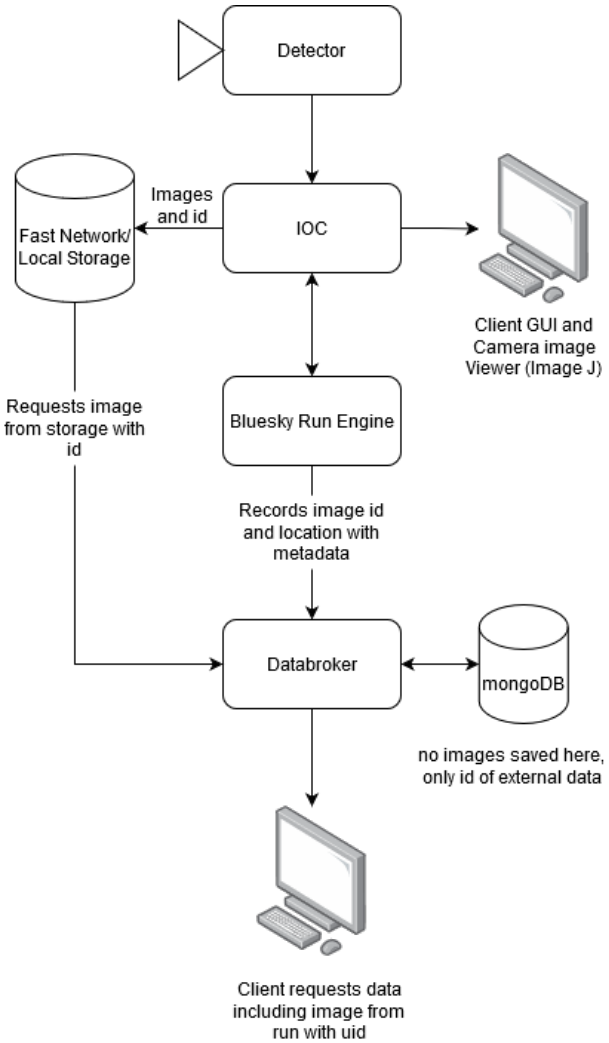


Figure 5: Accessing images by reference with Databroker.

Separating images from the metadata reduces the requirements for the mongoDB database storage and network.

Bluesky is critical to BESSY II achieving it's FAIR data objectives. Every measurement taken with Bluesky can also acquire values describing the state of the entire machine. The structuring of devices with Ophyd gives context to the names of parameters.

Callbacks can be defined which injest documents from the run engine. These callbacks allow metadata to be harvested automatically about samples, repetitive important information added automatically to eLogs, and data and metadata to be exported to publicly accessible repositories like ICAT in standardized Nexus format. This was explored in a case study at EMIL [3]. Notably because documents can be subscribed to over ZMQ, these processes can be separated from the critical task of measuring and saving experimental data.(See Fig. 4)

USER INTERFACES

To allow for simple overview representations synoptic screens will be needed showing positions within their limits, status information, health of devices. For these more engineering screens CS-Studios Phoebus will be used.

Interacting with Bluesky can currently be performed in three ways. Each has benefits and drawbacks, and each is prefered by different users. All are being supported and developed at BESSY II.

IPython is the most mature and popular means of interaction, and the most familiar for users of spec. It can easily and reliably produce live plots of acquired data. Browsing acquired data through the Databroker is more cumbersome than the other interfaces.

Jupyter notebooks can also be used. It has been challenging to get live plotting to work consistently in this environment. For our users who like using Jupyter notebooks, the benefits of the environment have outweighed this issue. The ability to create demonstrations which can be run and shared has been useful for training and collaboration.

The package Bluesky-Widgets provides collection of QT widgets for standard tasks like browsing and plotting from the Databroker, interacting or monitoring the Queueserver, or editing and running plans. These can be integrated with existing QT based GUI's, but at development at BESSY II is focusing on building on the demo application provided with the package. The application is an excellent base which can be adjusted to the needs of a particular user.

Initial deployment of Bluesky at EMIL used the IPython environment, and exported data to .spec files which could then be opened with PyMca [12]. This provided the easiest path to getting a working system that was familiar to beamline scientists.

INTEGRATION WITH MACHINE LEARNING TOOLS

An initial study was performed to investigate using a Reinforcement Learning (RL) agent with the Bluesky Run Engine (RE) with the aim of reducing harmonic orbit perturbations. An inhouse python package, called *Naus*, allowed for communication between the RE and the well established Tensorflow RL agent using ZMQ. Unfortunately this simple demonstrator set-up proved too slow to be used in production. Consequently developments of tweaked sub packages Bluesky-RL, Bluesky-Queueserver and Bluesky-adaptive are being followed now. Hopefully the *Naus* package can be refactored to take advantage of the Queueserver framework.

OUTLOOK

Considerable progress has been made in the last year, and the success of the work at EMIL and at the accelerator bodes well for the roll out to other beamlines. A team of 6 engineers across different departments is now working together to develop the BESSY II Bluesky ecosystem. Bluesky implementation and development moved to the core of a BESSY II reinforcement and modernization program. Visible achievements are convincing enough to ask for additional support of the funding agency.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR03

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

A route to making data acquired at the beamlines FAIR has been identified, and Bluesky has made it possible. Further work is needed to put this into production, not least making it easy for users to authenticate and associate their session with an investigation ID.

Authentication will also play a role in the development of access control at complex beamlines with shared resources like EMIL. Further work is required to understand how this can be incorporated with Bluesky.

The pathfinder study looking at integrating Bluesky with RL agents showed promise, but was not fast enough to be used in production in the accelerator. The work at other facilities on this topic is being followed closely.

The Bluesky user community continues to grow at labs of all sizes around the world. The energy and solution focused attitude of the people involved is one of the projects biggest assets.

ACKNOWLEDGEMENTS

The authors thank all HZB staff members active in maintaining the complex controls infrastructure at BESSY II for many fruitful and clarifying discussions. This includes beamline and instrument scientists, accelerator controls, beamline optics as well as central IT networking and storage.

Additionally we would like to thank the developers of the Bluesky ecosystem for their work and help with our many questions.

REFERENCES

[1] R. Müller, A.F. Balzer, P. Baumgärtel, G. Hartmann, O.-P. Sauer, and J. Viefhaus, "Modernization of Experimental Data Taking at BESSY II", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 65–69. doi:10.18429/JACoW-ICALEPCS2019-MOCPL02

[2] Bluesky Project, https://Blueskyproject.io/.

[3] G. Gerrit, "FAIR Meets EMIL: Principles in Practice", presented at ICALEPCS21, Shanghai, China, Oct, 2021, paper WEBL05, this conference.

[4] L. Vera Ramirez, "Machine Learning Tools Improve BESSY II Operation", presented at ICALEPCS21, Shanghai, China, Oct, 2021, paper THAL01, this conference.

[5] spec, Certified Scientific Software, https://certif.com

[6] https://Sardana-controls.org

[7] Pshell,Alexandre Gobbo, PSI, https://github.com/paulscherrerinstitute/pshell

[8] pytest is a framework that makes building simple and scalable tests easy https://docs.pytest.org/en/6.2.x/.

[9] server for queueing Bluesky plans, https://github.com/Bluesky/Bluesky-queueserver

[10] A tool for controlling 2D detectors, https://areadetector.github.io/master/index.html

[11] A data access tool for Bluesky, https://Blueskyproject.io/databroker/.

[12] X-ray fluoresence data analysis, http://pymca.sourceforge.net/.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR04

CONTINUOUS SCANS WITH POSITION BASED HARDWARE TRIGGERS

H. Enquist, A. Bartalesi, B. Bertrand, J. Forsberg, A. Freitas, V. Hardion, M. Lindberg,
C. Takahashi, MAX IV Laboratory, Lund, Sweden

Abstract

In traditional step scanning, repeated starting and stopping of motors leads to inefficient usage of the x-ray source. In addition to increasing measurement times, this also increases the risk of sample radiation damage. We have developed a system where scans are performed while continuously moving the motors. To ensure stable repeatable measurements, the detector triggers are generated, in hardware, from the motor encoder positions. Before the scan starts, a list of positions is generated. That is then used to generate the triggers when these positions are reached.

The solution is implemented with Tango and Sardana. The encoder signals from the motors are connected both to the IcePAP motion controller for closed loop operation, and a PandABox which is used as the trigger source.

The scan is controlled by a TriggerGate controller, that calculates the motor positions and configures the PandABox. The scanned motor can be either a single motor, for example a sample translation stage, or a combined motion like a monochromator. When combined motions are used, these are using the parametric trajectory mode of the IcePAP. This enables continuous scans of coupled axes with non-linear paths.

INTRODUCTION

Many experiments require performing scans to measure some quantity as a function of another. The easiest approach is to use a step scan, and therefore this is very commonly used. In a step scan each step consists of moving the scanned axis to a certain position, then arming and triggering the detector, waiting for acquisition to finish, and finally reading out the data. This process is repeated for every step. Only the time spent acquiring the data is useful, while the rest can be considered deadtime. The starting and stopping of the scan axis (typically one or several motors), and the arming of the detector can take a considerable amount of time. Under typical conditions, scans can run no faster than a few points per second. If the acquisition time is then in the millisecond range, the deadtime tends to make up for the majority of the time required to perform the scan.

An alternative solution is to keep the scan axis moving continuously during the scan. This has previously been implemented using combinations of software and hardware solutions [1-3]. In this case, the scanned axis needs only to be started and stopped once. The aqusition loop then consists of only triggering the detector, waiting for acquisition to finish, and readout. Many detectors are able to read out the data in millieconds, meaning the scan can run at hundreds of points per second. This is the approach taken in this work. We have chosen to implement a system that handles the motions and trigger generation completely in hardware.

CONTINUOUS MOTION

Letting the motion to run continuously during the scan is straightforward when the axis corresponds to a single motor that needs to run at constant speed. This is the case when for example scanning along the surface of a flat sample using a single motor. But in many cases the motion involves several motors, that can't simply be moved at constant speed. To scan for example the energy of a plane grating monochromator, both the mirror and grating positions must move acording to some formulas. They also need to perform their motions synchronized, in order to keep the x-ray beam stable on the sample, with the right energy and properties. As illustrated in Fig. 1, a simple software pseudomotor will perform motions that start and end at the correct positions, but while moving, the path does not follow the ideal trajectory.

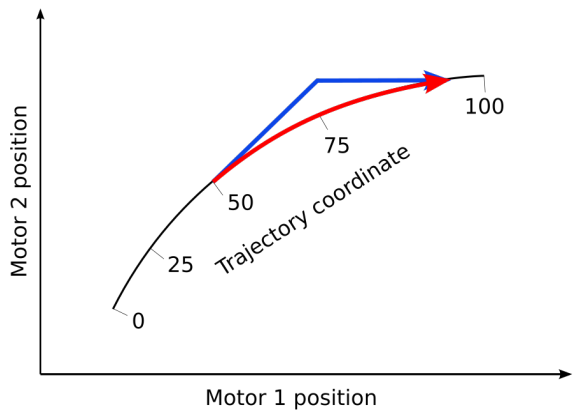


Figure 1: The difference between typical pseudomotions and parametric trajectories for a move along the parameter axis. The pseudomotor starts moving both motors at their nominal velocity. Once the first motor has reached it's position, the second continues moving towards the target. The parametric trajectory on the other hand follows the trajectory the whole way by continuously adjusting the motor velocities.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Parametric Trajectories

The non-linear motions and multi-motor synchronization can be solved by using a motor contoller that supports motions following parametric trajectories. In this work we have used the IcePAP system [4].

The parametric mode presents one or several motors as a single axis, with a single position attribute. The position can be in any unit, for example nm or eV. This is achieved by generating a table for each motor, of physical motor position in motor steps versus trajectory position in the desired unit. These tables are then uploaded to each driver. Once all tables are uploaded, the motors can be moved together in the trajectrory unit. When moving to a position between the points of the table, the IcePAP interpolates the individual motor positions from each table. Either linear or spline interpolation can be used, and the step size of the tables must be chosen to keep the maximum interpolation error within acceptable limits.

Time-based Trigger Generation

During the scan, the detector must be triggered as the scan axis reaches the desired positions. The easiest approach is to use time-based triggers. The triggers are then generated from the estimated times for when the scanned axis will reach each desired position. This works for any motor, without the need for extra cabling for encoder signals. The disadvantage is that there is some uncertainty of exactly when the motor will arrive at the desired positions, and the positions where the data was recorded may differ somewhat from the desired ones. This can be improved by capturing the motor positions at each trigger event. However this still can make comparing different scans challenging, since they were taken at slightly different positions.

Position-based Trigger Generation

Instead of time ased triggers, it's possible to monitor the motor positions, and generate the based on when the motors arrive at each position. This removes the uncertainty in the positions. In this work, we have used a PandABox [5] to generate the triggers. This device can support a very wide range of applications, and is easily configured by connecting and configuring functional blocks in a web-based gui. The blocks and connections between them are implemented in an FPGA. It can generate triggers based on numerous inputs, and is able to read standard motor encoder signals directly.

The employed configuration, or schema, for the PandABox is shown in Fig. 2. The "Sequencer" block of

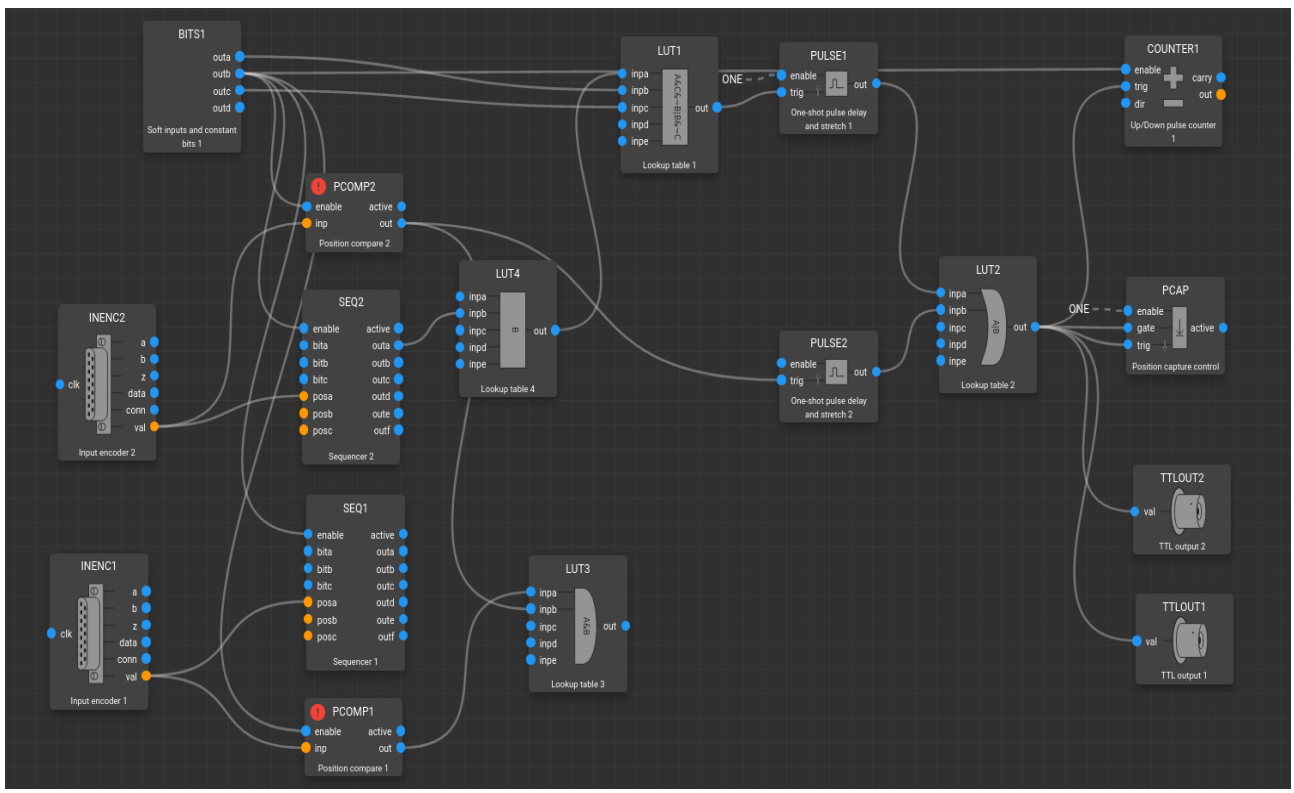


Figure 2: The PandABox schema. The encoder signals from the monochromator are connected to the INENC blocks. Currently the 2nd input, INENC2, is used for trigger generation. The sequencer block, SEQ2, is responsible for generating the triggers, based on a list of positions and the value from the encoder block. The output of the sequencer is connected to the TTL output via some logic blocks for selecting other operation modes, controlled via the BITS block. The COUNTER block keeps track of the generated triggers, and is useful for diagnostics. The PCAP block captures the values of selected variables when triggered. This is used to record the encoder signal values for both monochromator motors, as well as the undulator (not shown in the schema).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR04

the PandABox compares its input value against a list, and generates output pulses as each value is reached. The input is wired to the encoder input block, and the output is routed to a TTL output for the detector. Additionally it's wired to the "PositionCapture" block, which is configured to capture the encoder values. This is used to record the positions for the motors in the scan data.

The values of the sequencer table are calculated using the same formulas as for generating the parametric tables for the motor controller. The equidistant trajectory positions are calculated from the scan parameters.

SOFTWARE IMPLEMENTATION

The software for this project was developed as Sardana [6] controllers. Sardana is a scan framework based on Tango [7]. A total of three Sardana controllers were developed for this project.

- A TriggerGate that configures the PandABox to generate the expected timing. The timing system can be position based, if the scanned motor encoder is connected to the PandABox (the current implementation uses the monochromator mirror encoder), or time based, where the encoder is either not connected or there are no motors involved in the scan.
- A MotorController that enables motions using the parametric trajetory mode of the IcePAP. This is implemented to control the energy of a plane-grating moochromator via the mirror and grating angles.
- A CounterTimerController for reading out the captured encoder positions from the PandABox.

Procedure of a Scan

Before a scan can be started, the parametric trajectory mode of the monochromator is initialized. This procedure is automated in the controller. It generates a list of motor positions for the two motors, for a range of energies configured by properties. It then uploads these to the motor drivers, and moves the motors onto the trajectory at a chosen energy value. To avoid collisions, this energy value must correspond to motor positions near where the motors are curently standing.

When launching a continuous scan of the monochromator energy, the following steps are performed:

- Sardana: Pass scan parameters to the TriggerGate (TG) controller and all involved detectors.
- TG: The list of energies is created. These are equidistant values from the start to the end position, with one value per step.
- TG: The corresponding motor encoder positions for the mirror is calculated for each step.
- TG: This list is sent to the Sequencer block of the PandABox.
- Sardana: Calculates start and end positions based on the motor acceleration and deceleration times, to allow the motor to accelerate and decelerate outside of the scan range.
- Sardana: Move the motor to the start position.
- Sardana: Arm detector and TriggerGate controllers.
- Sardana: Sets the correct velocity of the monochromator motor in eV/s, and starts motion.
- TG: Each time the motor reaches the next position in the list it generates a trigger pulse.
- Sardana: While the scan is running Sardana polls the detectors for new data and appends the new points to it's record list. Depending on the speed the scan runs at, Sardana may receive zero, one, or several values from each detector per polling interval.
- Sardana: After the motion stops and all detectors finish their acquisition, Sardana saves the data to file.

Once the scan starts running, the motions and triggers are controlled completely in hardware by the IcePAP and PandABox. This allows scans to run at high rates without risk of missing or delayed triggers because of external factors such as network traffic or other load on the Sardana system. The role of Sardana during this stage of the scan is just to collect the recorded measurement values. Any delay that occurs during this loop will not affect the running scan.

This example did not include the undulator movement. For very short scans, the undulator can be left in one position if its bandwidth is sufficient to cover the scan range. Optionally, the undulator can also be moved during the continuous scan. In the present scope, the scans are limited to a small enough range to allow the undulator to move on a linear trajectory. The undulator is controlled by a dedicated IcePAP system. Sardana calculates and sets the velocity and acceleration time for this motion to match the scan parameters, and sends the command to start the motion to the motor controllers for monochromator and undulator at the same time. Because of possibly different network latencies this means the undulator and monochromator will not start at the exact same time. This can be overcome by letting one IcePAP trigger the motion of the other via a dedicated trigger cable. However, the relatively wide bandwidth of the undulator relaxes the mono-undulator synchronization requirements, and the uncertainty from the network falls well within acceptable limits.

MEASUREMENTS

A set of measurements were performed to validate the performance of the continuous scans against normal step scans. The measuerments were performed at the FlexPES beamline at MAX-IV Laboratory [8].

The sample was nitrogen, and the scan measured the absorption at the K-edge in the 400 – 402.5 eV range. The scans were done in steps of 0.01 eV, with 250 steps, a 100 ms exposure time and a 100ms latency time between points.

The data was collected using an Alba EM# electrometer [9].

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR04

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

For the particular measurement shown in Fig. 3, the step scan took 248 seconds, while the continuous took 57 seconds. The minimum possible time is 250*(0.1+0.1) = 50 seconds. The continuous scan comes close to this value, the extra time is spent on setting up all controllers, and the acceleration/deceleration phases of the motors.

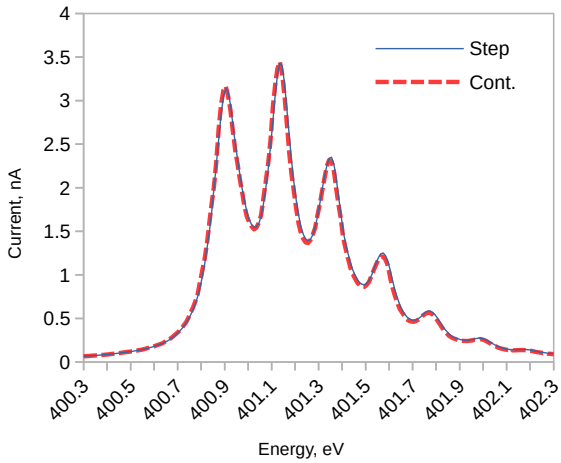


Figure 3: Comparison of two absorption scans of the Nitrogen K-edge. The solid blue line shows a standard step scan, the dashed red line shows a continuous scan.

For the step scan, the scan took 248 seconds, meaning that about 80% of the time was spent on moving to the next point. Figure 4 illustrates the differeces in how the time is spent during these two scans. Because only a minor part of the time during the step scan is spend on aquiring data, it cannot be made much faster by reducing the exposure time. If the exposure and latency times had been reduced by a factor 2, the step scan would have taken an estimated 223 seconds, while the continuous would have taken 32 seconds.

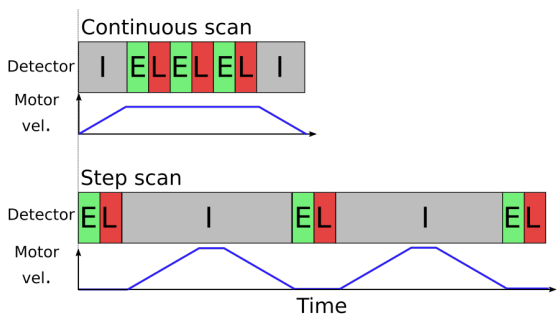


Figure 4: Illustration comparing how time is spent during a continuous scan and a step scan. For the detector, “E” denotes exposure, “L” latency (which also includes reading out the detector), and “I” idle. For clarity, this illustation shows a scan of only 3 steps.

FAST ACQUISITION CHALLENGES

Continuous scanning, especially when acquiriring at high rate, puts higher requirements on the used components than step scans.

Detectors

Fast continuous scans means that the used detectors must sample the signal at a high rate. This can cause issues if the bandwidth of the equipment is too low. The Alba EM# electrometer has an optional low-pass filter in the analog path which is useful for illustrating this issue. This filter can be configured for a 10 Hz cutoff. It is then tempting to use this filter to reduce the noise in the recorded signals when aquiring at any rate below 10 Hz. When aquiring at frequencies approaching 10 Hz, it is then logical to assume a slight smoothing of sharp features. But a low-pass filter changes the phase of the signal, meaning that the value captured at each trigger will have been delayed by the filter. This effectively leads to a shift of the data along the scan axis, where the amount of shift is difficult to predict. The solution is to ensure that the analog bandwidth of each instrument is sufficiently high to keep this effect within acceptable limits.

Motors

In case of sensitive equipment like a monochromator, the precision requirements are high already for step scans [10]. Adding the condition that these requirements must also be fulfilled while in motion leads to new challenges.

One is that the motions can create vibrations that may disturb sensitive equipment like monochromators. Vibrations tend to worsen with increased speed, and this may ultimately limit the achievable scanning speed for some systems.

Another issue arises if the motors rely on a closed feedback loop to reach the necessary accuracy. For step scans it can be tolerated that the position deviates while moving, as long as the position after the motion has stopped is accurate. But in continuous scans the closed loop must ensure that the motion closely tracks the target position during the entire motion. Careful tuning of the loop parameters can improve this. Lowering the scanning speed is also benefinial.

Best results are obtained for systems that were designed with continuous scans in mind. This means that vibrations are minimized, and the mechanical accuracy is high, to reduce the amount of corrections needed by the feedback loop.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRBR04

OUTLOOK

Performing continuous scans provides the obvious benefit that scans can be done considerably faster. This allows much more efficient use of the X-ray beam at storage ring facilities, where it often is the case that the signal level is strong enough to perform high quality measurements with short exposure times.

The increased speed leads to the added benefit that the sample is exposed to a much smaller radiation dose for for each scan. This is an important factor for for example biological samples that are often sensitive and easily damaged by exposure during scans.

The continous scans can be implemented for many types of measurements. Here we used it to scan the energy of a monochromator. A follow up project is planned to include the motion of the insertion device, also using the parametric trajectory mode of the IcePAP. This will allow scans over a larger energy range.

ACKNOWLEDGEMENTS

The authors thank the staff at the MAX IV FlexPES beamline for performing the measurements.

REFERENCES

[1] L. Joly *et al.,* "Fast continuous energy scan with dynamic coupling of the monochromator and undulator at the DEIMOS beamline", in *J. Synchrotron Rad.,* vol. 21, pp. 502-506, May 2014. doi:10.1107/S1600577514003671

[2] A. F. Balzer, E. Schierle, E. Suljoti, M. Witt, and R. Follath, "Status of the Continuous Mode Scan for Undulator Beamlines at BESSY II", in Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15), Melbourne, Australia, Oct. 2015, pp. 1091-1095. doi:10.18429/JACoW-ICALEPCS2015-THHA3O02

[3] Manuel Izquierdo *et al.*, "SUMS: synchronous undulator–monochromator scans at Synchrotron Soleil" in *J. Synchrotron Rad.,* vol. 19, pp. 619-626, June 2012. doi:/10.1107/S090904951201374X

[4] The IcePAP motion controller from ESRF, https://www.esrf.eu/Instrumentation/DetectorsAndElectronics/icepap

[5] PandABox, https://www.ohwr.org/project/pandabox/wikis/home

[6] SARDANA, http://www.sardana-controls.org

[7] TANGO, http://www.tango-controls.org

[8] FlexPES beamline at MAX IV Laboratory, https://www.maxiv.lu.se/accelerators-beamlines/beamlines/flexpes/

[9] J. A. Avila-Abellan et al., "Em# Electrometer Comes to Light", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 137-142. doi:10.18429/JACoW-ICALEPCS2017-TUAPL04

[10] P. Sjöblom, G. Todorescu and S. Urpelainen, "Understanding the mechanical limitations of the performance of soft X-ray monochromators at MAX IV laboratory", in *J. Synchrotron Rad.,* vol. 27, pp. 272-283, March 2020. doi:10.1107/S1600577520000843

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

List of Authors

Bold papercodes indicate primary authors; ~~crossed out~~ papercodes indicate 'no submission'

— A —

Abadie, L. **THAR01**
Abeillé, G. MOPV034, WEAR01
Abell, D.T. **WEPV023**, WEPV024
Abiven, Y.-M. FRBR01, ~~FRBR05~~, **~~FRXL05~~**
Adrianek, K. THBR02, THPV033
Afshar, N. **~~MOBR05~~**
Aghababyan, A. MOPV027, THPV025
Aghaei, Z. FRBL03
Ahmad Mehrabi, F. FRBL03
Airiau, J-P. **MOPV003**
Ajmal, M. TUBR03, TUPV019
Akbari, M. FRBL03
Akeroyd, F.A. **TUPV048**, WEAR03
Akinotcho, S. TUPV042
Alberti, V. FRAR01
Alfaurt, P. MOPV002
Aljamal, B.R. MOPV020
Alnajjar, D. THPV021, FRBL05
Althaus, A. WEPV007
Alves, D. WEPV019
Amador, I.A. TUPV031
Amelineau, L. TUPV042, **~~FRBR05~~**
Amjad, A. FRAR01
Amodio, A. THPV041
Andolfato, L. MOBL01
Andreassen, O.Ø. MOPV044
Aranha, P.D. TUPV003
Araujo, D.H.C. WEPV026, THPV021
Arena, G. TUBL05, TUPV016
Argomedo, J. MOBL01
Armanet, S. WEPV048
Arpaia, P. THPV041
Arruat, M. THBR02
Arruda, L.C. **WEPV034**, **THPV001**
Aryshev, A. ~~TUPV018~~
Asko, A. THPV013
Astrain, M. TUBR01
Augsburger, R.A. THBL02
Aurelius, O. ~~FRBR06~~
Awwal, A.A.S. MOPV021, **WEAL01**
Ayyagiri, N. TUPV013
Azzopardi, G. **WEPV016**, **THPV012**, **THPV040**

— B —

Bacchim Neto, F.A. WEPV034
Bacher, R. **MOPV005**
Baggiolini, V. TUPV047
Bai, D.P. WEBL02
Baker, K.R.L. **MOPV019**, FRBL01
Baker, K.V.L. TUPV048, **WEAR03**
Balana, A. MOPV002
Baldi, C. MOPV040
Baldwinson, B.E. ~~MOBR05~~
Banerjee, D. TUPV047
Barber, S.K. ~~THBR05~~
Barbour, A.M. FRBR02
Barczyk, A. ~~FRBL04~~
Bargueden, P. MOPV001
Barnes, A.I. MOAL02, MOAR02, **MOPV021**
Barreto, G.T. WEPV034, THPV001
Bartalesi, A. FRBR04
Bartolini, M. MOAL03
Bartoszek, P.R. ~~TUPV024~~, ~~WEPV030~~
Baudrenghien, P. THBR02
Bauvir, B. MOBL02, WEPV006
Beaulac, L. MOAL02, MOPV021
Becheri, F. MOPV037
Beglarian, A. TUPV011
Behere, A. FRAL05
Bell, P.J. MOAL01
Bellato, M.A. TUBL05
Belohrad, D. TUPV037
Benes, N. MOBL01
Bengtsson, J. ~~TUPV008~~
Bernardi, M.L. FRBL05
Bernhard, J. TUPV047
Bertrand, B. MOPV034, WEAR01, **THBL01**, **THPV011**, FRBR04
Bez, E. TUPV039
Bhasker, A. **MOPV047**
Bheesette, S. TUPV013
Bielawski, B. THPV033
Birke, T. ~~TUPV008~~, THAL01
Bisiach, D. **THPV009**
Blackwell, B. MOPV021, WEAR02
Blanch-Torné, S. MOPV037, TUBL03
Blanchard, S. **TUPV031**
Blanco Viñuela, E. **MOPV010**, TUPV035, WEBR02, WEPV042, **~~FRXL04~~**
Blomley, E. ~~TUBR05~~
Bode, W.A. TUBL02
Bohm, C. TUAL03
Bondar, V. MOBL04, ~~TUPV044~~
Bonnay, P. TUPV006
Borrego, J.H.P.D.C. TUPV035
Bortolato, D. TUBL05, TUPV016
Boukabache, H. MOBR01, WEBR01
Bourgoin, C.B. ~~FRBR05~~
Bourtembourg, R. MOPV034, TUBL03, WEAR01

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Bouvel, S. TUPV042, FRBR01
Brandl, G. WEAR01
Braun, T. MOPV034, WEAR01
Bravin, E. WEPV044
Brede, K. THPV025
Bridger, A. TUBL02
Briquez, F. FRBR01
Brito Neto, J.L. TUPV004, **WEPV001**, WEPV003, THPV021
Brown, K.A. THAL04
Brugger, M. TUPV047
Bruhwiler, D.L. WEPV024
Bruhwiler, K. WEPV022, THAL04
Brunton, G.K. MOAL02, MOAR02, MOPV021, WEAR02
Bräger, M. TUBL01, TUPV005
Bräuning, H. TUPV009
Budge, T.S. WEAL01
Bueno, C.S.N.C. **WEPV002**
Buonomo, B. THAL03
Burdzanowski, L. MOPV043
Burger, S. WEPV044
Burns, K. MOPV021
Burridge, R.A. FRBL01
Butterworth, A.C. THBR02, THPV033
Büttner, M. MOPV039
Buttu, M. FRAL02
Bär, M. WEBL05

— C —

Calcanha, M.P. WEPV034, THPV001
Calder, S. WEPV022
Calia, A. WEPV019
Camacho, L.U. WEPV034
Camps Gimenez, A. THPV010
Canova, H.F. WEPV034, THPV001
Canzari, M. FRAR01
Cao, J.S. ~~TUPV045~~
Capovilla, L.G. WEPV002
Carannante, G. THAR01
Carboni, G. FRAL02
Cardelli, F. THAL03
Cardoso, F.H. WEPV026, WEPV028, WEPV034
Cargnelutti, M. THPV009
Carinan, C. ~~TUPV044~~
Carlin, E.G. WEPV024, THAL04, ~~THPV017~~, FRBR02
Carpenter, A. WEPV025, **THPV043**
Carrasco, S.A. THBL02
Carver, L.R. MOPV012
Casey, A.D. MOAL02
Castro Morales, J.R. MOAL02, MOPV021
Cazorla, R. THPV010
Ceesay-Seitz, K. MOBR01, **WEBR01**
Celary, M. MOPV034, ~~**THPV008**~~
Chaize, J.M. MOPV012
Chandra, P. THAR03
Charitonidis, N. TUPV047
Charrondière, C. **MOPV044**, **TUPV032**
Chatzigeorgiou, N. TUPV031
Chen, G. **TUPV028**
Chen, G.H. TUBR04
Chen, J.F. TUBR04
Chen, Y.X. WEPV012
Chevallay, E. TUPV032
Chilingaryan, S.A. TUPV011
Chiozzi, G. **MOBL01**
Chiriac, C.C. MOPV045
Chmielewski, K. ~~TUPV024~~, ~~WEPV030~~
Cho, S.Y. TUPV027, **WEPV015**, THPV037
Cho, W.S. **MOPV035**
Chu, C.P. ~~TUPV045~~
Cinquegrana, P. WEAL02
Ciuffetti, P. THAL03
Ciupinski, M.A. THPV048
Clark, R.D. MOPV047
Clark, S.L. **MOPV049**
Clift, M. ~~MOBR05~~
Clot, R. **WEPV036**
Cobb, T.M. TUPV040, THBL04
Coelho, E.P. TUPV004
Cole, L. TUPV048
Coleman, S.J. ~~THBR05~~
Cook, N.M. ~~**THBR05**~~, **FRBR02**
Copy, B. MOPV010, TUBL01, TUPV005
Corne, M. MOPV002
Corruble, D.C. TUPV042, ~~FRBR05~~
Cortey, H. **FRAL01**
Cosic, D.D. **WEPV005**
Costa, A. ~~**THPV045**~~
Costa, I. **THPV010**
Costa, R. MOBL04, ~~TUPV044~~
Costa, V. TUBR01
Cotte, M. ~~THAR02~~
Cseppentö, L. **MOPV039**
Cuim, F.M.O. TUPV025
Cuní, G. **MOPV037**

— D —

Da Silva, J. FRBR01
da Silva, T.P. THBR01
Dam, M. TUPV034
Dang, J.J. WEPV015
Daniluk, G. THBR02
Danzeca, S. MOPV010
Darde, D. TUPV007, MOPV001
Daudin, L. **MOPV002**
Davis, B.A. **MOBL05**
Davoine, L. TUPV036, THPV049
Day, D.S. TUPV009
de Albuquerque, G.S. **WEPV003**

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

de Assis Schmidt, G. WEBR02
De Biaggi, M. FRAL02
de la Cruz, G. MOPV045
de Martel, J.-B. **MOBL03**
De Nolf, W. ~~THAR02~~
Deghaye, S. MOPV041, TUPV033
Degtiarenko, P. THPV043
Del Nero, F.A. MOBR03
Delamare, Ch. MOPV010
Delfs, T. MOPV005
Denis, V. MOPV003
Derrez, C.S. TUPV015
Deshpande, J.A. TUPV013
Desmarchelier, G. MOPV001, WEPV031
Di Calafiori, D.R.S. TUAR01
Di Carlo, M. **TUBL04**
Di Castro, M. THPV012
Di Giovenale, D. THAL03
Di Giulio, C. THAL03
Diaw, A. WEPV022
Diaz Cano, C. MOBL01
Ding, J.G. TUBR04, WEPV040
Ding, L. ~~**THKL01**~~
Dissertori, G. TUAR01
Dixon, J. MOAL02, MOAR02
Djambazov, L. TUAR01
Do Carmo, L.P. TUPV004, THPV021
Dolci, M. TUBL04
Donatti, M.M. **WEPV026**, WEPV028
Donon, Y. THPV041
Du, Y. WEPV024
Duhme, H.T. THPV025
Dunne, K.E. TUAL03
Dunning, D.J. WEPV020, WEPV021
Durandeau, H. MOPV003
Duteil, G. TUPV006
Dyer, P.S. MOPV049

— E —

Edelen, A.L. THAL02
Edelen, J.P. **WEPV022**, WEPV023, WEPV024, **THAL04**, ~~THBR05~~, ~~**THPV017**~~
Egli, J. THBR02, THPV033
Eguiraun, M. MOAL01, **FRAR01**, ~~FRBR06~~
Ehsan, W. MOBL04, ~~TUPV044~~
Elangovan, Y. TUPV013
Elkaim, E. FRBR01
Elliot, R.A. MOPV026
Elson, P.J. **MOPV040**
Emery, J. TUPV037
Engblom, C. **TUPV042**, ~~FRBR05~~
Enomoto, Y. THPV028
Enquist, H. **FRBR04**
Eriksson, T. MOAL01
Esenov, S.G. MOBL04, ~~TUPV044~~
Estes, C.M. MOAL02, MOAR02
Evans, G.G. TUPV025

— F —

Fabbri, R. MOBL04, ~~TUPV044~~
Falcon-Torres, C. MOPV037
Fallejo, R.N. MOPV047
Fara, A. FRAL02
Farnham, B. TUPV005
Farvacque, L. MOPV012
Fedorov, M. **MOAL02**, MOAR02, MOPV021, WEAR02
Felice, H. TUPV034
Fenner, M. THPV025
Fernandes, R.N. **WEPV048**
Fernández Adiego, B. MOPV042, **WEBR02**, **WEPV042**, ~~FRXL04~~
Fernández Maltas, T. THPV010
Fernández, J. **WEPV031**
Fernandez, R.P. TUPV025
Ferrand, G. MOPV001, WEPV031
Ferreira, R. TUPV031
Ferro, G. WEPV006
Finch, I.D. MOPV019, **WEPV049**, ~~WEPV050~~, ~~THPV016~~
Fiorina, D. ~~**TUPV017**~~
Fisher, S. ~~**THAR02**~~
Fishler, B.T. MOBL05
Fiszer, L. MOPV041
Fitzek, J. MOPV013, WEPV047
Flayol, M. MOPV002
Flegel, M.S. MOAL02, MOAR02, WEAR02
Flucke, G. **MOBL04**, ~~TUPV044~~
Foggetta, L.G. THAL03
Forsberg, J. THPV011, FRAR01, FRBR04
Fourtillan, P. MOPV003
Franca, J.V.B. WEPV034, THPV001
Francisco Rebelo, J.D. TUPV031
Franco, J.G.R.S. WEPV027
Frank, W.A. **MOPV020**
Freeman, B.G. WEPV023
Freitas, Á. MOAL01, THBL01, FRBR04
Freitas, G.F. WEPV027
Fröhlich, L. **MOPV027**
Fujii, Y. WEPV037
Fujimaki, M. MOPV015, WEPV038
Fukui, T. MOPV014
Fukunishi, N. MOPV015, WEPV038
Fukushima, K. WEBL04
Furukawa, K. THPV028
Furusato, F.S. FRBL05

— G —

Gabourin, S. **WEPV041**
Gabriel, M. TUPV047

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Gaget, A. MOPV001, MOPV024, **THPV022**
Gaio, G. **WEAL02**, **WEPV008**
Galatas, E. **THPV013**
Galeas, P. THBL02
Gandor, M. MOPV033
Gargiulo, F. THPV041
Gatignon, L. TUPV047
Gayet, Ph. FRAL03
Gelain, F. TUBL05, TUPV016
Georg, J. TUPV012
Geraldes, R.R. **TUPV004**, WEPV001, WEPV002, WEPV003
Gerber, N. MOBR01
Gerbershagen, A. TUPV047
Ghribi, A. TUPV006
Giacchini, M.G. **TUPV015**
Gil, P. WEPV031
Giles, A. WEPV024
Gilfellon, A.J. WEPV021
Gill, J.R. THBR02
Gingold, T. THBR01, THBR02
Giovanetti, G. MOBL04, ~~TUPV044~~
Godambe, S.V. THAR03
Goeries, D. MOBL04, ~~TUPV044~~
Gohier, F. MOPV001, MOPV024, TUPV007, WEPV031
Golonka, P. **THPV049**
Gomes, A. TUPV025
Gomes, P. TUPV031
Gong, G.H. THBR03
Gonzalez Berges, M. ~~**FRXL03**~~
González Cobas, J.D. THBR02
Gonzalez, A. ~~FRBR06~~
Gonzalez-Aguilera, J.P. THAL02
Gonzalez-Berges, M. THPV042
Gopalan, V.K. MOAL02, **MOAR02**
Goralczyk, L.G. **MOPV017**
Gorbonosov, R. MOBL03, TUPV033, TUPV047
Goryl, P.P. MOPV033, MOPV034, **TUBL03**, WEAR01
Gostkin, M. TUAR03
Götz, A. MOPV012, MOPV034, TUBL03, **WEAR01**
Gougnaud, F. **MOPV001**, MOPV024, TUPV007
Gras, J-J. THPV042
Grech, L. **WEPV019**
Greve, N. WEBL05
Griffin, N.L. TUPV050
Grunewald, S. MOPV027
Gu, G. THBR03
Guedes, L.C. WEPV002, THPV021
Günther, G. **WEBL05**
Guerrero, A. **TUPV037**
Guiho, P. MOPV001, TUPV007
Guijarro, M. ~~**TUPV043**~~
Guo, Y.H. **MOPV030**, MOPV031, WEPV013, THPV006
Gurriana, L. TUPV025

— **H** —

Hackel, B. MOAR02
Haedar, H. FRBL03
Hagmann, G. THBR02, THPV033
Halastra, S.E. TUBL01
Hall, C.C. WEPV022, THAL04
Hamada, Y. MOPV014
Hamanaka, M. WEPV038
Harding, P. TUBL04
Hardion, V. **MOAL01**, TUBL03, WEAR01, THBL01, FRAR01, FRBR04, ~~**FRXL02**~~
Hariharan, J. FRAL05
Harper, J.R. TUPV048, TUPV049, ~~WEBL03~~
Hartmann, G. THAL01
Hauf, S. MOBL04, ~~TUPV044~~
Havart, F. WEPV042
Haziot, A. TUPV034
He, Y.C. WEAL03, ~~**TUPV021**~~
Heerey, S. MOAL02
Hensler, O. MOPV027
Hickin, D.G. MOBL04, ~~TUPV044~~
Hierholzer, M. TUPV012
Higurashi, Y. MOPV015
Himmerlich, M. TUPV039
Hisano, A. **WEPV010**
Hoffmann, C.M. WEPV022
Hoffmann, D. ~~TUBR05~~
Hoffstadt Urrutia, A. MOBL01
Hogan, R.B. ~~MOBR05~~
Hoguin, F. **TUPV033**
Holloway, P. TUPV050
Hong, T.M. ~~**THBR04**~~
Horita, A.Y. **MOBR03**, TUPV001, TUPV004
Hosoda, N. **MOPV014**
Hostettler, M. WEPV019
Howells, G.D. MOPV019, WEPV049, ~~**WEPV050**~~
Hrabia, M. TUPV047, WEPV018
Huang, Z.Y. **WEPV039**
Huenupan, F. THBL02
Hüser, S. WEPV007
Hüther, H.C. MOPV013, WEPV047
Hulek, W.K. TUPV036
Hurdelbrink, U. THPV025
Husson, A.A. MOPV002

— **I** —

Iancu, V. TUPV028

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Iftekharuddin, K.M. WEPV025, THPV043
Iitsuka, Y. THPV027
Ikezawa, E. MOPV015
Iman, I. FRBL03
Imao, H. MOPV015
Iodice, L. THPV041
Irannejad, F. MOPV041
Isakov, H. MOPV001
Ishii, M. MOPV014
Iwasaki, M. WEPV010

— J —

Jackson, S. TUPV037, WEPV019, WEPV044, THPV014
Jafarzadeh, M. FRBL03
Jagudin, E. ~~FRBR06~~
Jakobsen, S. THPV048
Jamilkowski, J.P. ~~THPV018~~
Jankowski, P.D. TUPV034
Janneck, J.W. ~~FRBR06~~
Jastrow, U. MOPV027
Jensen, S. THPV014
Ji, H.F. THPV047
Jiang, Y.X. MOPV016, FRAR03
Jiang, Z.Y. THBR03
Jiao, Y. THPV047
Jignesh, J. WEBR05
Jiménez Estupiñán, R. TUAR01
Jin, D.P. WEAL03, WEBR03
Joannem, T.J. MOPV001, **TUPV007**
Johansen, E. ~~TUAL01~~
Johnsen, C. ~~FRBL04~~
Jones, M.D. ~~WEBL03~~
Joubert, A.F. MOPV034, **MOPV036**, TUBL02, WEAR01
Joy, S. FRAL05
Juerges, T. **MOAR03**, ~~MOPV046~~
Junkes, H. **WEBL01**
Justus, M. THPV031

— K —

Kaji, H. **THPV027**, THPV028
Kalantari, B. ~~**TUAL01**~~
Kalsi, S.S. THAR01
Kamigaito, O. MOPV015
Kamikubota, N. THPV029
Kammering, R. MOPV027, ~~**THPV034**~~
Kamoshida, A. MOPV015
Kampmeyer, C.K. TUPV012
Kaneko, K. MOPV015
Karentzos, E. **MOPV009**
Karkinsky, D.A. **WEBR05**
Karlovsek, P. MOBL02
Kasemir, K.-U. MOAR01
Kay, H. THPV025
Kazarov, A.G. ~~**TUAR02**~~
Kazarski, S. FRBR03
Kazimi, R. WEPV023
Kedron, K. MOPV034
Keilman, M.V. WEPV024, ~~THPV017~~
Keller, H. MOPV027
Keymer, D.P. TUPV048
Khaleghi, A. **FRBL03**
Khan, S. WEPV007
Kidera, M. MOPV015
Kikuzawa, N. THPV029
Killenberg, M. **TUPV012**
Kim, J.H. TUPV027, WEPV015, **THPV037**
Kim, S.W. **THPV038**
Kim, Y.K. THAL02
King, J.C. TUPV048, **TUPV049**
Kiyomichi, A. MOPV014
Klimovskaia, A. MOBL04, ~~TUPV044~~
Klingberg, J.M. MOAL01
Knap, G. TUPV040, **THBL04**
Knap, M. MOBL02
Kobets, V.V. MOPV018, TUAR03
Kocevar, H. WEPV048
Kocharyan, V. MOPV027
Kofukuda, L.M. TUPV003
Komiyama, M. MOPV015, **WEPV038**
Koning, D.J. MOAR02, WEAR02
Koning, P. WEAR02
Kontogiorgos, G.N. MOBR03, **TUPV001**, **TUPV002**, TUPV003, WEPV002
Kopmann, A. TUPV011
Koprek, W. ~~TUAL01~~
Kornweibel, N. MOBL01
Kostopoulos, A.F. MOPV017
Kostrzewa, K. ~~TUPV024~~, ~~WEPV030~~
Kovari, Zs. **THPV015**
Kowalski, G.W. ~~**TUPV023**~~
Kowalski, T. ~~TUPV024~~, ~~WEPV030~~
Koyama, R. MOPV015
Kozak, T. TUPV012
Koziol, P.J. TUPV034
Krahl, R. WEBL05
Krasna, J. **MOBR04**
Krecic, S. WEAL02, WEPV008
Krepp, S. **MOPV013**
Krieger, M. WEBL01
Krimm, A. TUPV009
Krmpotić, L. THPV031
Kruk, G. THPV015
Ku, K.H. THPV038
Kubin, M. WEBL05
Kuhn, A. FRBL05
Kumagai, K. MOPV015, WEPV038
Kuner, B. MOPV026
Kuntzsch, M. TUPV010, THPV031
Kuzmanović, P. THBR02, THPV033

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

— L —

Lachacinski, B. MOPV002
Lacoste, D. MOPV012, WEAR01
Lacuata, R. MOAL02, MOPV021, **WEAR02**
Ladzinski, T. WEPV042
Laface, E. THPV011
Lampater, U. MOBL01
Lampridis, D. **THBR01**, THBR02
Landoni, M. FRAL02
Lange, R. **MOPV026**, **TUBR01**
Langston, M.H. **MOPV048**
Lauener, J. MOPV041, TUPV033
Lay, S.C. TUPV050
Leach, R.R. MOPV021, WEAL01
Leban, P. THPV009
Leclercq, N. MOPV012, WEAR01
Ledeul, A. **MOPV045**
Lee, S. THPV037
Lee, S. TUAL03
Lee, S. MOBL02
Lee, W.R. **MOBL02**
Lee, W.W. THPV038
Legat, U. MOBR04, THPV031
Lein, A. MOBL04, ~~TUPV044~~
Leluan, B. ~~FRBR05~~
Lena, F.R. WEPV001
Lestrade, A. ~~FRBR05~~
Lethin, R. MOPV048
Letourneau, P.D. MOPV048
Leveneur, M. WEBR01
Li, C. WEPV011, WEPV039
Li, J. ~~TUPV008~~
Li, J.S. **WEPV012**
Li, J.Y. THPV047
Li, L. MOPV032, **THPV007**, THPV036
Li, M. TUBR04
Li, S. MOPV016, FRAR03
Li, W. WEPV011
Li, X. TUAL02
Li, Y.L. FRAR01
Liebermann, H. WEPV047
Lilley, S. FRBL01
Lilli, G. TUBL05
Lima, C.V. TUPV031
Lima, G. ~~FRBR06~~
Lin, C. ~~**FRKL01**~~
Lindberg, M. MOAL01, FRBR04
Lindhé, I. ~~**FRBR06**~~
Lipiński, M.M. **MOPV011**, THBR02
Lippek, H. THPV025
Liszcz, M. **MOPV034**, TUBL03, WEAR01
Liu, G. TUPV046, WEPV011, WEPV039
Liu, H.T. MOPV030
Liu, J. MOPV032
Liu, S.H. TUAL02
Liu, X.M. WEBL02
Liu, Y. TUAL02
Liu, Y.J. TUBR04
Liuzzo, S.M. **MOPV012**
Löhnert, T. TUPV048, TUPV049, WEAR03
Long, A.J. TUPV048, TUPV049
Long, W. **TUAL02**
Lonza, M. ~~FRXL03~~
López Sánchez, A. MOPV034
Lopez-Miguel, I.D. MOPV042, WEPV042
Losito, R. MOPV010
Lotode, A. MOPV001
Lotrus, P. MOPV024, TUPV007
Loustalet, N. MOPV003
Lowe-Webb, R.R. WEAL01
Lu, X.H. **THPV047**
Ludwig, M. TUPV005
Luiz, S.A.L. TUPV004
Lunt, S.A. **TUPV030**
Luo, J. MOPV032, THPV007, **THPV036**
Lustermann, W. TUAR01
Lv, H.H. **WEBL02**
Lynch, J. WEPV024

— M —

Madisa, K. TUBL02
Madysa, N. MOBL03
Maheshwari, M. WEPV020
Mahmoudi, K. FRBL03
Mahoney, K.L. MOAR01
Maia, L.G. MOBL04
Majid, A. TUBR03, TUPV019, **TUPV020**
Majumder, G. TUPV013
Makowski, D.R. MOBL02, THAR01
Maldonado, J. WEPV024
Malinowski, S.M. ~~THPV016~~
Malka, J. ~~TUPV044~~
Mamchyk, D. MOBL04, ~~TUPV044~~
Mannix, O. WEBL05
Mantion, P. **MOPV041**, TUPV033
Marcato, D. **TUBL05**, TUPV016
Marchese, L. **TUAR01**
Mariette, Y. MOPV001
Markowski, P. ~~TUPV024~~, ~~WEPV030~~
Marques Oliveira, T. **TUBL01**
Marqueta, A. WEBR05
Marsching, S. ~~**TUBR05**~~
Martin Anido, D. TUBL01
Martinelli, V. TUBL05, TUPV016, THAL03
Martins dos Santos, L. TUPV001, **TUPV003**,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

WEPV002
Martins, F. **TUPV025**
Martins, J.P.S. TUPV015
Maruta, T. WEBL04
Masi, A. MOPV010
Matei, C. TUPV028
Matej, Z. MOAL01, ~~**FRBL04**~~, ~~FRBR06~~
Mathes, D. MOPV005
Mathisen, D.G. MOPV021
Matli, E. MOPV010, THPV013
Mayes, C.E. THAL02
Mazur, P. THAR01
Mazzucco, E. TUPV014
McDonald, R.J. MOBL05
Medina, D. ~~**WEPV043**~~
Medley, S.A. ~~**THPV016**~~
Meirose, B. ~~THPV039~~
Melkumyan, D. MOBR02
Meng, C. THPV047
Mercado, R. **TUPV050**
Mertens, T. ~~TUPV008~~
Mexner, W. ~~TUBR05~~
Meyer, J.M. ~~THAR02~~
Mi, Q.R. TUBR04
Miao, H.F. TUBR04
Miccolis, M. MOAL03
Michalik, D. THBR01
Migoni, C. FRAL02
Miller Kamm, V.J. MOAL02, MOPV021, WEAL01
Milosic, T. TUPV009
Minashkin, V.F. MOPV018
Miqueles, E.X. FRBL05
Miyahara, F. THPV028
Mkrtchyan, T. ~~TUAR02~~
Moazzam, Y. THBL04
Moeller, P. WEPV024, ~~THPV017~~, FRBR02
Mohanan, S. THAR03, FRAL05
Mol, J.J.D. MOAR03, ~~MOPV046~~
Mommertz, M. MOPV027
Monnereau, G. MOPV001
Monteiro, P. TUPV042
Montis, M. TUPV015, THPV004, THPV005
Moraes, M.A.L. MOBR03, TUPV001, TUPV002, TUPV003, TUPV004, WEPV001, WEPV002, WEPV003, THPV021
Moreno, G.B.Z.L. WEPV002, WEPV026
Moreno, J. TUBR01
Moreton-Smith, C. TUPV048
Morgado, J.B. TUBL04
Moschovakos, P. ~~**TUPV038**~~
Moser, D.G. WEPV023
Mostafa, J. **TUPV011**
Motzkau, H. TUAL03
Moura, F.N. WEPV034
Müller, A.-S. ~~TUBR05~~
Müller, R. THAL01, FRBR03
Mueller, R. MOPV013, **WEPV047**
Munaron, E. TUBL05, TUPV016, **THPV004**
Munoz, L.E. TUPV042, **FRBR01**
Murach, T. MOBR02
Muralidhar, S. MOPV021

— N —

Nabywaniec, M. **MOPV033**
Nadot, V. MOPV001, **MOPV024**
Nagahara, H. WEPV010
Nagatomo, T. MOPV015
Nagler, R. WEPV024, ~~THBR05~~, ~~THPV017~~, FRBR02
Nair, P.M. TUPV013
Nakagawa, T. MOPV015
Nakamura, T. WEPV038
Nakano, T. WEPV010
Nakashima, Y. WEPV010
Nakayoshi, K. **WEPV037**
Nallin, P.H. **WEPV027**
Nan, J. ~~FRBR06~~
Nash, B. **WEPV024**, FRBR02
Navidpour, P. FRBL03
Nawaz, D.A. TUBR03, **TUPV019**, TUPV020
Neema, S.K. **FRAL05**
Nemesure, S. MOPV049
Nes, T.H. TUPV034
Neto, A. THAR01
Ng, A. ~~MOBR05~~
Nguyen, L.M. FRAR01
Ni, Z. ~~MOPV008~~, **MOPV032**, THPV007, THPV036
Nienhaus, M. FRBL05
Nikiel, P.P. ~~TUPV038~~
Nilsson, J.M.C. ~~**WEBL03**~~
Ning, D.J. ~~**WEKL01**~~
Nishi, T. MOPV015
Nogiec, J.M. **WEPV033**
Noll, D. WEPV018
Nordt, A. WEPV041
Noureddine, A. FRBR01
Novel González, S. THBR02, THPV033
Nozdrin, M.A. **MOPV018**, TUAR03
Nozik, A.A. ~~**TUPV029**~~
Nunes, I. THAR01
Nutter, B.J. ~~FRXL05~~

— O —

Ockards, M.T. MOPV036
Ohki, T.O. MOPV015

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Ohnishi, J. MOPV015
Okada, K. MOPV014
Ondreka, D. TUPV009, WEPV047
Oppermann, P. WEBL01
Oram, D.E. TUPV048, TUPV049, WEAR03
Orlati, A. FRAL02
Oskarsson, M. ~~THAR02~~
Ostrega, M. **THPV048**
Ostroumov, P.N. WEBL04
Otto, Th. WEBR02
Oulevey, T. THBL03, ~~**FRXL01**~~
Oven, Ž. **THPV031**
Oya, I. MOBR02
Oyamada, K. MOPV015
Ozeki, K. MOPV015

— P —

Pacheu, V. MOPV021
Padmanabhan, S. MOAL01
Paglovec, P. THPV009
Palaha, A.S. **TUPV040**
Palluel, J.P. THBR02
Pan, Y. MOPV021
Panchumarti, J. THAR01
Panyam, N. TUPV013
Papageorgiou Koufidis, A TUBL01
Paraskevopoulos, C. **MOPV006**
Parenti, A. MOBL04, ~~TUPV044~~
Pascual-Izarra, C. MOPV034, MOPV037, WEAR01
Patankar, S. MOPV021, WEAL01
Patel, B.P. MOPV021, WEAL01
Paul, M. MOAL02, MOPV021
Pavinato, S. WEPV041
Pedersen, U.K. THBL04
Peixinho, A.Z. FRBL05
Pellegrin, F. MOBL01
Peng, Y.M. THPV047
Perek, P. MOBL02, THAR01
Perissinotto, L.S. **WEPV028**
Perna, A.V. WEPV003
Perrin, D. MOBR01, WEBR01
Perry, A. MOPV001
Peryt, M. TUPV047, **WEPV018**
Peters, F. MOPV027
Petersson, J.E. ~~**THPV039**~~
Petrosyan, A. MOPV027
Petrosyan, G. MOPV027
Petrosyan, L.P. MOPV027
Petrosyan, V. MOPV027
Pezzetti, M. **THPV041**, **FRAL03**
Piccino Neto, A.C. WEPV001, WEPV002
Pichoff, N. WEPV031
Piermarini, G. THAL03
Pigny, G. TUPV031
Pilliaud, B. TUPV042
Pinches, S.D. THAR01
Pincot, F.O. TUPV034
Pinty, V. FRBR01
Pioli, S. **THAL03**
Piton, J.R. WEPV002, **THPV021**
Pivetta, L. MOPV034, TUBL03, WEAR01
Plastun, A.S. WEBL04
Pogorelov, I.V. WEPV024
Polack, F. ~~FRBR05~~
Pollard, A.E. **WEPV020**, **WEPV021**
Pollet, P. ~~TUAL01~~
Polli, R.W. WEPV027
Poncet, F. MOPV012
Pons, J.L. **MOPV025**
Pons, X. THPV048
Popov, K. ~~**TUPV018**~~
Poppi, S. **FRAL02**
Poscia, A. THBR01
Pranovi, L. THPV004, **THPV005**
Previero, T. MOPV010
Previtali, G. MOBL04
Prieto Diaz, I. WEBR05
Punna, M. **TUPV013**, **THAR03**

— R —

Rae, B. **TUPV047**
Rahman, M. WEPV025, THPV043
Rahne, L. THPV009
Rai, B. TUPV048
Rakitin, M.S. WEPV024, FRBR02
Ramirez, F. TUPV028
Ramirez, J.G. WEPV031
Rao, B. MOPV016, FRAR03
Reascos Portilla, A.K. TUPV039
Redaelli, S. THPV012
Rees, N.P. MOAL03
Regnell, S. WEPV048
Rehlich, K. MOPV027
Reinfeld, E. MOPV001
Reszela, Z. MOPV037
Rey, A. THPV033
Reymond, H. **TUPV034**
Reynolds, C.J. THBL04
Řežende, J.H. TUPV003
Ribeiro, B. TUBL04
Ribeiro, H.R. FRAR01
Ricci, G. THPV040
Richter, S.C. TUPV034
Richter, T.S. ~~WEBL03~~
Ries, M. ~~TUPV008~~, THAL01
Righini, S. FRAL02
Rivilla, D. TUBR01
Rizzi, M. THBR02
Roberts, P.J. TUPV050
Rocha, A.P. TUPV031
Roche, S.T. ~~THBR04~~

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Roderick, C. TUPV047, THPV013
Rodrigues, G.L.M.P. WEPV034
Roetta, M. TUBL05, TUPV016
Roger, A. TUPV007
Rojec, U. THPV031
Romagnoli, G. TUPV047
Romanovschi, M. MOPV019, ~~THPV016~~, FRBL01
Roncarolo, F. TUPV037
Roque, C.S.B.N. TUPV002
Rose, S.C.F. **~~MOPV038~~**
Rosenqvist, J.T.K. FRAR01
Rothe, D. TUPV012
Roussel, R.J. **THAL02**
Rozenshteyn, R. MOPV021
Rubio-Manrique, S. MOPV034, MOPV037, WEAR01
Ruiz, M. TUBR01
Rumiz, L. TUPV014
Rybka, D. ~~TUPV024~~, ~~WEPV030~~
Rybnikov, V. MOPV027

— S —

Saad, M. FRAR01
Sakamoto, N. MOPV015
Sakashita, K. WEPV037
Salnikov, A. MOAL01, ~~FRBL04~~
Salvachua, B. WEPV016, THPV012
Salvat, D. THPV010
Samantas, A. **THPV042**
Sanchez, R.J. MOPV021
Sanfelici, L. WEPV026
Santander-Vela, J. **MOAL03**
Sanz, D. TUBR01
Saoulis, A.A. MOPV019, WEPV049, **FRBL01**
Saqib, N.U. **TUBR03**, TUPV019, TUPV020, **THPV046**
Sargsyan, L. MOPV050
Satake, I. WEPV010
Sato, K.C. THPV029
Sato, M. THPV028
Satoh, M. WEPV010
Sauter, S. MOPV021
Savarese, G. TUBL05, **TUPV016**
Saveri Silva, M. TUPV003, WEPV001, WEPV003
Scalamera, G. WEAL02
Schälicke, A. THAL01
Schaller, A. MOPV013, WEPV047
Schilling, M. MOBL01
Schirmer, D. **WEPV007**
Schlarb, H. THPV025
Schlenker, S. ~~TUPV038~~
Schlesselmann, G. MOPV027
Schlögl, R. WEBL01
Schmidt, T. MOBR02
Schnizer, P. **~~TUPV008~~**, THAL01, FRBR03
Schoefer, V. THAL04
Schofield, B. MOPV017, TUPV005, **TUPV035**, ~~FRXL04~~
Schüngel, T. WEPV007
Schuh, M. ~~TUBR05~~
Schurmann, J. ~~FRBR06~~
Schwanke, U. **MOBR02**
Schwarz, A. THPV031
Schwinn, A. TUPV009
Sciacca, E. ~~THPV045~~
Scrivens, R. WEPV018
Sebdaoui, M. ~~FRBR05~~
Sedghi, B. MOBL01
Segalla, L.F. TUPV001
Seguel, R.S. THBL02
Segura, G. MOPV045
Semissatto, G.T. WEPV026
Sepulveda, A. THBL02
Serans, M.H. THBR01
Shehzad, N. TUPV012
Shen, G. **~~FRZL02~~**
Shen, T.C. **THBL02**
Sher, F. TUBR03, TUPV019, TUPV020, THPV046
Shmuely, I. MOPV001
Shroff, K. **~~FRAR04~~**
Shukla, M.R. THBR01
Sierra, R. MOPV010
Sierra-Maíllo Martínez, D. MOPV044
Silenzi, A. MOBL04, **~~TUPV044~~**
Silva, G.G. MOBR03
Silva, V. MOPV001
Silverstein, S.B. TUAL03
Silvola, R.P.I. **MOPV050**
Simrock, S. THAR01
Sinkarenko, I. MOPV040
Sitek, M. ~~TUPV024~~, ~~WEPV030~~
Sjöblom, P. MOAL01
Škabar, M. THPV009
Skovhede, K. ~~FRBL04~~
Smith, W. WEBL05, **FRBR03**
Snijder, T. MOAR03, **~~MOPV046~~**
Soares Augusto, J.A. TUPV025
Solenne, N. MOPV001, TUPV007
Solfaroli Camillocci, M. THPV012
Solomon, Y. MOPV001
Sommer, H. MOBL01
Song, Y.G. **TUPV027**, WEPV015, THPV037
Souza, M.S. WEPV003
Spengler, G. MOBR02
Speroni, R. WEBR02
Spierer, A. THBR02, THPV033
Spina, T.V. **FRBL05**

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Spruce, D.P. MOAL01, MOBL04, ~~TUPV044~~
Sridharan, P. TUPV013, THAR03, FRAL05
Srivastava, S. TUPV013, FRAL05
Steinbrück, R. TUPV010
Steinhagen, R.J. **TUPV009**
Stockinger, E. **TUPV005**
Strangolino, G. WEAL02, **FRAR02**
Suarez Valles, M. MOBL01
Suda, K. MOPV015
Sugimoto, T. MOPV014
Sugimura, H. THPV028
Sukhanov, A. ~~**THPV018**~~
Suleiman, R. WEPV025, THPV043
Sumiński, M. THBR02, **THPV033**
Sun, X.K. WEPV039
Swart, P.S. TUBL02
Szewiński, J. ~~**TUPV024**~~, ~~**WEPV030**~~
Sznajd, J. MOPV045
Szuba, J. MOBL04, ~~TUPV044~~

— T —

Tagg, J. MOPV044, **TUPV039**
Tajima, Y. THPV029
Tak, T.H. MOBL02
Takahashi, C. FRBR04
Takemura, N. WEPV010
Tamim, N. MOPV001
Tamura, M. MOPV015
Tanvir, A. TUBR03
Tao, F. **WEBR04**
Taranowski, M. MOPV021
Tarkeshian, R. TUPV015
Taurel, E.T. MOPV012
Teichmann, M. MOBL04
Teixeira, D.I. **TUPV036**
Tempel, T. MOPV005
Tennant, C. WEPV023, WEPV025, THPV043
Terunuma, N. ~~TUPV018~~
Theidel, G. ~~TUAL01~~
Thibaux, G. TUPV042
Thompson, P. WEPV033
Timm, J.H.K. TUPV012
Tlusty, D. ~~**MOPV022**~~
Tolentino, H.C.N. TUPV003, WEPV002
Topaloudis, A. **WEPV044**
Torrent, P. MOPV003
Tournier, J-C. MOPV017, **MOPV042**, WEPV042
Townsend, S.L. MOAR02, WEAR02
Trevi, M. **TUPV014**
Trifonov, A. MOPV018, **TUAR03**
Tripaldi, F. WEAL02, WEPV008
Trovò, M. WEAL02
Trudel, A.H. **TUPV006**
Trunschke, A. WEBL01
Tsalas, M. THAR01
Tsotskolauri, P. **THPV032**
Tucker, D. MOAR02, MOPV021
Tura, Q. TUPV006
Turner, D.L. WEPV025, THPV043
Turturica, G. TUPV028
Twum, S.N. **TUBL02**
Tzanis, P. **FRAL04**
Tzvetkov, S. MOPV024

— U —

Uchida, K. ~~THPV008~~
Uchiyama, A. **MOPV015**, WEPV038
Urakawa, J. ~~TUPV018~~
Urbaniec, B. **MOPV043**

— V —

Vadilonga, S. WEBL05, FRBR03
Vaga, F. ~~FRXL01~~
Vaher, J.L. MOAR02
Valdes Santurio, E. **TUAL03**
Valentino, G. WEPV016, WEPV019, THPV012
Valléau, M. FRBR01
Valls, V. ~~THAR02~~
Van Herck, W. WEBR05, **WEPV006**
van Tilborg, J. ~~THBR05~~
Van Wonterghem, B.M. MOAL02
Vanden Eynden, M. THBL03
Varghese, G. TUPV012
Varlec, J. MOBR04
Velotti, F.M. WEPV044
Venter, A.J. TUBL02, WEAR01, ~~**FRZL01**~~
Vera Ramiréz, L. **THAL01**, FRBR03
Verlaat, B. TUPV030
Vestin, K. MOPV026
Veyrunes, E. WEPV044
Vićentijević, M. WEPV005
Vidal, J.O. ~~FRXL04~~
Vidyaratne, L.S. **WEPV025**, THPV043
Vigali, A. THPV009
Vinter, B. ~~FRBL04~~
Vitello, F.R. FRAL02
Vittor, D. TUPV014
Vodopivec, K. MOAR01
Voelkle, M. MOPV041
Voirin, R. **THBL03**
Volpe, L.M. WEPV001
Vrcic, S. **TUBR02**

— W —

Wagner, P. MOBR02
Wai, S. MOPV036
Walter, A. WEPV024
Walter, A. MOPV013, WEPV047
Wang, B.J. MOPV030, MOPV031,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

	WEPV013, THPV006
Wang, D.	**THPV028**
Wang, D.Y.	WEAL03
Wang, H.Y.	**TUBR04**
Wang, J.	WEPV012
Wang, J.	WEAR02
Wang, J.G.	WEPV039
Wang, L.	WEAL03
Wang, R.	MOPV030, **MOPV031**, WEPV013, THPV006
Wang, W.	FRAR03
Wang, W.S.	MOPV016
Washington, R.A.	MOPV020
Watanabe, T.	MOPV015
Watanabe, Y.	MOPV015
Weaver, S.	MOAL02
Webb, S.D.	WEPV024
Weber, H.	WEBL01
Wei, J.	MOPV048
Weiss, G.	THPV011, ~~FRAR04~~
Weninger, C.	~~FRBL04~~
Wenninger, J.	WEPV019
White, K.S.	**MOAR01**
White, S.M.	MOPV012
Wiegart, L.	FRBR02
Wilgen, J.	MOPV027
Wilhelmsen, K.C.	MOPV021, WEAL01
Wilks, R.	WEBL05
Wilksen, T.	MOPV005, MOPV027, **THPV025**
Williams, D.C.	MOAR01
Willner, C.	TUPV012
Wilson, B.A.	MOPV021
Winkler, C.	MOPV026
Winter, A.	MOBL02
Włostowski, T.	**THBR02**
Wojciechowski, Z.	~~TUPV024~~, ~~WEPV030~~
Wong, D.	~~MOBR05~~
Wrona, K.	MOBL04, ~~TUPV044~~
Wu, F.Y.	**MOPV016**, FRAR03
Wu, X.	~~TUPV021~~, WEAL03
Wujek, A.	THBR02

— X —

Xie, N.	MOPV030, MOPV031, WEPV013, **THPV006**
Xie, X.H.	MOPV016, **FRAR03**
Xie, Z.X.	WEAL03
Xu, G.	THPV047
Xu, S.	WEPV039
Xu, W.	WEPV011
Xuan, K.	WEPV011, WEPV039
Xue, K.	WEAL03

— Y —

Yadav, A.	**MOBR01**
Yamada, K.	MOPV015
Yamauchi, H.	MOPV015
Yan, Y.B.	THBR03, ~~**FRZL03**~~
Yang, F.	WEPV012
Yang, M.	**THPV029**
Yang, Y.	MOPV016, FRAR03
Yao, Ms.	~~**MOPV008**~~
Ye, Q.	THPV047
Ye, Y.M.	THBR03
Yilmaz, U.	TUBL04
Yin, J.	~~**TUKL01**~~
Youngman, C.	MOBL04, ~~TUPV044~~
Yu, C.L.	TUBR04, **WEPV040**
Yu, P.X.	**THBR03**
Yu, Y.B.	**WEPV011**
Yun, S.P.	THPV037
Yusa, Y.A.	MOPV015

— Z —

Žagar, A.	MOBL02
Zambon, L.	WEAL02, FRAR02
Zanzottera, S.	**THPV014**, THPV042
Zchut, T.	MOPV001
Zeng, L.	WEAL03
Zenker, K.	**TUPV010**, THPV031
Zhang, H.	MOPV021
Zhang, M.	MOPV016, FRAR03
Zhang, P.L.	MOPV016, FRAR03
Zhang, S.	TUPV042
Zhang, T.	**WEBL04**
Zhang, W.	~~**MOPV007**~~
Zhang, Y.	~~**TUPV045**~~
Zhang, Y.L.	~~TUPV021~~, WEAL03, **WEBR03**
Zhang, Z.	**TUPV046**
Zhao, H.	WEPV040
Zhao, H.	WEBL02
Zhao, L.	THBR03
Zhao, Q.	WEBL04
Zhao, Y.L.	THPV047
Zhao, Z.T.	~~**MOKL01**~~
Zhemchugov, A.	TUAR03
Zheng, H.	WEPV012
Zheng, W.	MOPV016, FRAR03
Zhou, X.	~~MOPV008~~, MOPV032, THPV036
Zhu, P.	**WEAL03**, WEBR03
Zhuravlyov, P.P.	TUAR03
Zilliox, T.	MOPV044, TUPV032
Zimny, M.Z.	THPV049
Zimoch, D.	MOPV026
Zwalinski, L.	TUPV030, TUPV036, THPV049
Zygaropoulos, L.	TUPV031
Żytniak, Ł.	MOPV033, ~~TUPV023~~

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Institutes List

7S
Peligros (Granada), Spain
- Fernández, J.
- Gil, P.
- Ramirez, J.G.

Aalborg University
Aalborg, Denmark
- Michalik, D.

Aalto University, School of Science and Technology
Aalto, Finland
- Stockinger, E.

ALBA-CELLS Synchrotron
Cerdanyola del Vallès, Spain
- Becheri, F.
- Blanch-Torné, S.
- Camps Gimenez, A.
- Cazorla, R.
- Costa, I.
- Cuní, G.
- Falcon-Torres, C.
- Fernández Maltas, T.
- López Sánchez, A.
- Pascual-Izarra, C.
- Reszela, Z.
- Rubio-Manrique, S.
- Salvat, D.

ALMA Observatory
Santiago, Chile
- Sepulveda, A.
- Shen, T.C.

ANL
Lemont, Illinois, USA
- Shen, G.

ANSL
Yerevan, Armenia
- Sargsyan, L.

Aquenos GmbH
Baden-Baden, Germany
- Marsching, S.

AS - ANSTO
Clayton, Australia
- Afshar, N.
- Baldwinson, B.E.
- Hogan, R.B.
- Ng, A.
- Wong, D.

ASCo
Clayton, Victoria, Australia
- Clift, M.

ASTRON
Dwingeloo, The Netherlands
- Juerges, T.
- Mol, J.J.D.
- Snijder, T.

BARC
Trombay, Mumbai, India
- Ayyagiri, N.
- Behere, A.
- Deshpande, J.A.
- Joy, S.
- Mohanan, S.
- Nair, P.M.
- Neema, S.K.
- Punna, M.
- Sridharan, P.
- Srivastava, S.

BESSY GmbH
Berlin, Germany
- Kuner, B.

Bhabha Atomic Research Centre (BARC)
Mumbai, India
- Chandra, P.
- Godambe, S.V.
- Hariharan, J.

BNL
Upton, New York, USA
- Barbour, A.M.
- Brown, K.A.
- Clark, S.L.
- Du, Y.
- Dyer, P.S.
- Giles, A.
- Jamilkowski, J.P.
- Lynch, J.
- Maldonado, J.
- Nemesure, S.
- Rakitin, M.S.
- Schoefer, V.
- Shroff, K.
- Sukhanov, A.
- Walter, A.
- Wiegart, L.

byte physics e.K.
Berlin, Germany
- Braun, T.

CAEP
Sichuan, People's Republic of China
- Li, L.
- Liu, J.
- Luo, J.
- Ni, Z.
- Yao, Ms.
- Zhou, X.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Catalyst IT
Wellington, New Zealand
- Harding, P.

CEA
LE BARP cedex, France
- Airiau, J-P.
- Clot, R.
- Cortey, H.
- Denis, V.
- Durandeau, H.
- Fourtillan, P.
- Loustalet, N.
- Torrent, P.

CEA LIST
Palaiseau, France
- Zchut, T.

CEA, DES-ISAS-DM2S, Université Paris-Saclay
Gif-sur-Yvette, France
- Darde, D.

CEA-DRF-IRFU
France
- Bargueden, P.
- Desmarchelier, G.
- Gaget, A.
- Gougnaud, F.
- Guiho, P.
- Lotode, A.
- Mariette, Y.
- Nadot, V.
- Roger, A.
- Solenne, N.

CEA-IRFU
Gif-sur-Yvette, France
- Darde, D.
- Ferrand, G.
- Gaget, A.
- Gohier, F.
- Gougnaud, F.
- Joannem, T.J.
- Lotrus, P.
- Monnereau, G.
- Nadot, V.
- Pichoff, N.
- Silva, V.
- Tzvetkov, S.

CEA/INAC
Grenoble Cedex 9, France
- Bonnay, P.

CENBG
Gradignan, France
- Alfaurt, P.
- Balana, A.
- Corne, M.
- Daudin, L.
- Flayol, M.
- Husson, A.A.
- Lachacinski, B.

CERN
Geneva 23, Switzerland
- Adrianek, K.
- Alves, D.
- Amador, I.A.
- Amodio, A.
- Andreassen, O.Ø.
- Arruat, M.
- Asko, A.
- Azzopardi, G.
- Baggiolini, V.
- Baldi, C.
- Banerjee, D.
- Baudrenghien, P.
- Belohrad, D.
- Bernhard, J.
- Bez, E.
- Bielawski, B.
- Blanchard, S.
- Blanco Viñuela, E.
- Borrego, J.H.P.D.C.
- Boukabache, H.
- Bravin, E.
- Brugger, M.
- Bräger, M.
- Burdzanowski, L.
- Burger, S.
- Butterworth, A.C.
- Büttner, M.
- Calia, A.
- Ceesay-Seitz, K.
- Charitonidis, N.
- Charrondière, C.
- Chatzigeorgiou, N.
- Chevallay, E.
- Chiriac, C.C.
- Ciupinski, M.A.
- Copy, B.
- Cseppentö , L.
- da Silva, T.P.
- Dam, M.
- Daniluk, G.
- Danzeca, S.
- Davoine, L.
- de Assis Schmidt, G.
- de la Cruz, G.
- de Martel, J.-B.
- Deghaye, S.
- Delamare, Ch.
- Di Castro, M.
- Donon, Y.
- Egli, J.
- Elson, P.J.
- Emery, J.
- Farnham, B.
- Fernández Adiego, B.
- Ferreira, R.
- Fiszer, L.
- Francisco Rebelo, J.D.
- Gabriel, M.
- Galatas, E.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Gayet, Ph.
- Gerber, N.
- Gerbershagen, A.
- Gill, J.R.
- Gingold, T.
- Golonka, P.
- Gomes, P.
- Gonzalez Berges, M.
- González Cobas, J.D.
- Gonzalez-Berges, M.
- Goralczyk, L.G.
- Gorbonosov, R.
- Gras, J-J.
- Grech, L.
- Guerrero, A.
- Hagmann, G.
- Halastra, S.E.
- Havart, F.
- Haziot, A.
- Himmerlich, M.
- Hoguin, F.
- Hostettler, M.
- Hrabia, M.
- Hulek, W.K.
- Iodice, L.
- Irannejad, F.
- Jackson, S.
- Jakobsen, S.
- Jankowski, P.D.
- Jensen, S.
- Karentzos, E.
- Kostopoulos, A.F.
- Kovari, Zs.
- Koziol, P.J.
- Kruk, G.
- Kuzmanović, P.
- Ladzinski, T.
- Lampridis, D.
- Lauener, J.
- Ledeul, A.
- Leveneur, M.
- Lima, C.V.
- Lipiński, M.M.
- Lopez-Miguel, I.D.
- Losito, R.
- Ludwig, M.
- Madysa, N.
- Mantion, P.
- Marques Oliveira, T.
- Martin Anido, D.
- Masi, A.
- Matli, E.
- Medina, D.
- Moschovakos, P.
- Nes, T.H.
- Nikiel, P.P.
- Noll, D.
- Novel González, S.
- Ostrega, M.
- Otto, Th.
- Oulevey, T.
- Palluel, J.P.
- Papageorgiou Koufidis, A.
- Perrin, D.
- Peryt, M.
- Pezzetti, M.
- Pigny, G.
- Pincot, F.O.
- Pons, X.
- Poscia, A.
- Previero, T.
- Rae, B.
- Reascos Portilla, A.K.
- Redaelli, S.
- Rey, A.
- Reymond, H.
- Richter, S.C.
- Rizzi, M.
- Rocha, A.P.
- Roderick, C.
- Romagnoli, G.
- Roncarolo, F.
- Salvachua, B.
- Samantas, A.
- Schlenker, S.
- Schofield, B.
- Scrivens, R.
- Segura, G.
- Serans, M.H.
- Shukla, M.R.
- Sierra, R.
- Sierra-Maíllo Martínez, D.
- Silvola, R.P.I.
- Sinkarenko, I.
- Solfaroli Camillocci, M.
- Speroni, R.
- Spierer, A.
- Sumiński, M.
- Sznajd, J.
- Tagg, J.
- Topaloudis, A.
- Tournier, J-C.
- Urbaniec, B.
- Vaga, F.
- Vanden Eynden, M.
- Velotti, F.M.
- Verlaat, B.
- Veyrunes, E.
- Vidal, J.O.
- Voelkle, M.
- Voirin, R.
- Wenninger, J.
- Włostowski, T.
- Wujek, A.
- Yadav, A.
- Zanzottera, S.
- Zilliox, T.
- Zimny, M.Z.
- Zwalinski, L.
- Zygaropoulos, L.

CNPEM
Campinas, SP, Brazil

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

- Bacchim Neto, F.A.
- Freitas, G.F.
- Moura, F.N.

Cosylab
Ljubljana, Slovenia
- Krmpotić, L.
- Legat, U.
- Oven, Ž.
- Rojec, U.

COSYLAB Japan
Ibaraki, Japan
- Uchida, K.

COSYLAB, Control System Laboratory
Ljubljana, Slovenia
- Karlovsek, P.
- Knap, M.
- Krasna, J.
- Varlec, J.

Creighton University
Omaha, NE, USA
- Tlusty, D.

CTA
Heidelberg, Germany
- Oya, I.

DELTA
Dortmund, Germany
- Althaus, A.
- Hüser, S.
- Khan, S.
- Schirmer, D.
- Schüngel, T.

DESY
Hamburg, Germany
- Aghababyan, A.
- Bacher, R.
- Brede, K.
- Delfs, T.
- Duhme, H.T.
- Fenner, M.
- Fröhlich, L.
- Georg, J.
- Grunewald, S.
- Hensler, O.
- Hierholzer, M.
- Hurdelbrink, U.
- Jastrow, U.
- Kammering, R.
- Kampmeyer, C.K.
- Kay, H.
- Keller, H.
- Killenberg, M.
- Kocharyan, V.
- Kozak, T.
- Lippek, H.
- Mathes, D.
- Mommertz, M.
- Peters, F.
- Petrosyan, A.
- Petrosyan, G.
- Petrosyan, L.P.
- Petrosyan, V.
- Rehlich, K.
- Rothe, D.
- Rybnikov, V.
- Schlarb, H.
- Schlesselmann, G.
- Shehzad, N.
- Tempel, T.
- Timm, J.H.K.
- Varghese, G.
- Wilgen, J.
- Wilksen, T.
- Willner, C.

DESY Zeuthen
Zeuthen, Germany
- Melkumyan, D.
- Murach, T.
- Schmidt, T.
- Wagner, P.

DLS
Oxfordshire, United Kingdom
- Cobb, T.M.
- Griffin, N.L.
- Holloway, P.
- Knap, G.
- Lay, S.C.
- Mercado, R.
- Moazzam, Y.
- Nutter, B.J.
- Palaha, A.S.
- Pedersen, U.K.
- Reynolds, C.J.
- Roberts, P.J.

DNSC
Dongguan, People's Republic of China
- Liu, Y.
- Wang, D.Y.

EFOR
Levallois Perret, France
- Bouvel, S.

EJIT
Hitachi, Ibaraki, Japan
- Iitsuka, Y.

Elettra-Sincrotrone Trieste S.C.p.A.
Basovizza, Italy
- Cinquegrana, P.
- Gaio, G.
- Krecic, S.
- Lonza, M.
- Mazzucco, E.
- Pivetta, L.
- Rumiz, L.
- Scalamera, G.
- Strangolino, G.
- Trevi, M.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Tripaldi, F.
- Trovò, M.
- Vittor, D.
- Zambon, L.

ESO
Garching bei Muenchen, Germany
- Andolfato, L.
- Argomedo, J.
- Benes, N.
- Chiozzi, G.
- Diaz Cano, C.
- Hoffstadt Urrutia, A.
- Kornweibel, N.
- Lampater, U.
- Pellegrin, F.
- Schilling, M.
- Sedghi, B.
- Sommer, H.
- Suarez Valles, M.

ESRF
Grenoble, France
- Bourtembourg, R.
- Carver, L.R.
- Chaize, J.M.
- Cotte, M.
- De Nolf, W.
- Farvacque, L.
- Fisher, S.
- Götz, A.
- Guijarro, M.
- Lacoste, D.
- Leclercq, N.
- Liuzzo, S.M.
- Meyer, J.M.
- Oskarsson, M.
- Poncet, F.
- Pons, J.L.
- Taurel, E.T.
- Valls, V.
- White, S.M.

ESS
Lund, Sweden
- Armanet, S.
- Derrez, C.S.
- Elliot, R.A.
- Fernandes, R.N.
- Gabourin, S.
- Kocevar, H.
- Laface, E.
- Martins, J.P.S.
- Nilsson, J.M.C.
- Nordt, A.
- Pavinato, S.
- Regnell, S.
- Richter, T.S.
- Rose, S.C.F.
- Tarkeshian, R.
- Vestin, K.
- Weiss, G.

ETH
Zurich, Switzerland
- Di Calafiori, D.R.S.
- Dissertori, G.
- Djambazov, L.
- Jiménez Estupiñán, R.
- Lustermann, W.
- Marchese, L.

EuXFEL
Schenefeld, Germany
- Bondar, V.
- Carinan, C.
- Costa, R.
- Ehsan, W.
- Esenov, S.G.
- Fabbri, R.
- Flucke, G.
- Giovanetti, G.
- Goeries, D.
- Hauf, S.
- Hickin, D.G.
- Klimovskaia, A.
- Lein, A.
- Maia, L.G.
- Malka, J.
- Mamchyk, D.
- Parenti, A.
- Previtali, G.
- Silenzi, A.
- Szuba, J.
- Teichmann, M.
- Wrona, K.
- Youngman, C.

F4E
Barcelona, Spain
- Neto, A.

FAU
Erlangen, Germany
- Krieger, M.
- Weber, H.

FCUL
Lisboa, Portugal
- Soares Augusto, J.A.

Fermilab
Batavia, Illinois, USA
- Nogiec, J.M.
- Thompson, P.

FHI
Berlin, Germany
- Junkes, H.
- Oppermann, P.
- Schlögl, R.
- Trunschke, A.

FRIB
East Lansing, Michigan, USA
- Fukushima, K.
- Maruta, T.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

- Ostroumov, P.N.
- Plastun, A.S.
- Zhang, T.
- Zhao, Q.

FYSIKUM, AlbaNova, Stockholm University
Stockholm, Sweden
- Dunne, K.E.
- Lee, S.
- Valdes Santurio, E.

GANIL
Caen, France
- Duteil, G.
- Ghribi, A.
- Trudel, A.H.
- Tura, Q.

GMV
Madrid, Spain
- Moreno, J.
- Sanz, D.

GRIT
Aveiro, Portugal
- Morgado, J.B.
- Ribeiro, B.

GSI
Darmstadt, Germany
- Bräuning, H.
- Day, D.S.
- Fitzek, J.
- Hüther, H.C.
- Krepp, S.
- Krimm, A.
- Liebermann, H.
- Milosic, T.
- Mueller, R.
- Ondreka, D.
- Schaller, A.
- Schwinn, A.
- Steinhagen, R.J.
- Walter, A.

HU Berlin
Berlin, Germany
- Schwanke, U.
- Spengler, G.

Huazhong University of Science and Technology, State Key Laboratory of Advanced Electromagnetic Engineering and Technology,
Hubei, People's Republic of China
- Li, S.
- Rao, B.
- Yang, Y.
- Zhang, M.
- Zhang, P.L.
- Zheng, W.

HUST
Wuhan, People's Republic of China
- Jiang, Y.X.
- Wang, W.S.
- Wang, W.
- Wu, F.Y.
- Xie, X.H.

HZB
Berlin, Germany
- Bengtsson, J.
- Birke, T.
- Bär, M.
- Greve, N.
- Günther, G.
- Hartmann, G.
- Kazarski, S.
- Krahl, R.
- Kubin, M.
- Li, J.
- Mannix, O.
- Mertens, T.
- Müller, R.
- Ries, M.
- Schälicke, A.
- Schnizer, P.
- Smith, W.
- Vadilonga, S.
- Vera Ramiréz, L.
- Wilks, R.
- Winkler, C.

HZDR
Dresden, Germany
- Justus, M.
- Kuntzsch, M.
- Schwarz, A.
- Steinbrück, R.
- Zenker, K.

I-Tech
Solkan, Slovenia
- Bisiach, D.
- Cargnelutti, M.
- Leban, P.
- Paglovec, P.
- Rahne, L.
- Škabar, M.
- Vigali, A.

IFIN-HH
Bucharest - Magurele, Romania
- Chen, G.
- Iancu, V.
- Matei, C.
- Ramirez, F.
- Turturica, G.

IHEP
Beijing, People's Republic of China
- Cao, J.S.
- He, Y.C.
- Ji, H.F.
- Jiao, Y.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Jin, D.P.
- Li, J.Y.
- Li, X.
- Liu, S.H.
- Long, W.
- Meng, C.
- Peng, Y.M.
- Xu, G.
- Ye, Q.
- Zeng, L.
- Zhang, Y.L.
- Zhang, Y.
- Zhao, Y.L.
- Zhu, P.

IHEP CSNS
Guangdong Province, People's Republic of China
- Lu, X.H.
- Wang, L.
- Wu, X.
- Xie, Z.X.
- Xue, K.

IKIU
Qazvin, Iran
- Aghaei, Z.
- Haedar, H.
- Iman, I.
- Khaleghi, A.
- Mahmoudi, K.

ILSF
Tehran, Iran
- Ahmad Mehrabi, F.
- Akbari, M.
- Jafarzadeh, M.
- Khaleghi, A.
- Navidpour, P.

IMP/CAS
Lanzhou, People's Republic of China
- Chen, Y.X.
- Guo, Y.H.
- Li, J.S.
- Liu, H.T.
- Wang, B.J.
- Wang, J.
- Wang, R.
- Xie, N.
- Yang, F.
- Zhang, W.
- Zheng, H.

INAF - IRA
Bologna, Italy
- De Biaggi, M.
- Orlati, A.
- Righini, S.

INAF - OA Teramo
Teramo, Italy
- Dolci, M.

INAF - OAAB
Teramo, Italy
- Canzari, M.
- Di Carlo, M.

INAF - OAC
Selargius (CA), Italy
- Buttu, M.
- Carboni, G.
- Fara, A.
- Migoni, C.
- Poppi, S.

INAF IRA
Bologna, Italy
- Vitello, F.R.

INAF-OACT
Catania, Italy
- Costa, A.
- Sciacca, E.

INAF-OAT
Trieste, Italy
- Alberti, V.

INAF-Osservatorio Astronomico di Brera
Merate, Italy
- Landoni, M.

INFN- Sez. di Padova
Padova, Italy
- Bellato, M.A.

INFN/LNF
Frascati, Italy
- Cardelli, F.
- Ciuffetti, P.

INFN/LNL
Legnaro (PD), Italy
- Arena, G.
- Bortolato, D.
- Gelain, F.
- Giacchini, M.G.
- Lilli, G.
- Marcato, D.
- Martinelli, V.
- Montis, M.
- Munaron, E.
- Pranovi, L.
- Roetta, M.
- Savarese, G.

Institute of High Energy Physics, CAS
Guangdong, People's Republic of China
- He, Y.C.

ITER Organization
St. Paul lez Durance, France
- Abadie, L.
- Bauvir, B.
- Carannante, G.
- Ferro, G.
- Jignesh, J.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

- Karkinsky, D.A.
- Lange, R.
- Lee, W.R.
- Marqueta, A.
- Nunes, I.
- Panchumarti, J.
- Pinches, S.D.
- Prieto Diaz, I.
- Simrock, S.
- Tak, T.H.
- Tsalas, M.
- Van Herck, W.
- Žagar, A.

J-PARC, KEK & JAEA
Ibaraki-ken, Japan
- Sato, K.C.

JAEA/J-PARC
Tokai-mura, Japan
- Kikuzawa, N.

JASRI
Hyogo, Japan
- Hamada, Y.
- Hosoda, N.
- Ishii, M.
- Kiyomichi, A.
- Okada, K.
- Sugimoto, T.

JINR
Dubna, Moscow Region, Russia
- Gostkin, M.
- Kobets, V.V.
- Minashkin, V.F.
- Nozdrin, M.A.
- Trifonov, A.
- Zhemchugov, A.
- Zhuravlyov, P.P.

JLab
Newport News, Virginia, USA
- Carpenter, A.
- Degtiarenko, P.
- Freeman, B.G.
- Kazimi, R.
- Moser, D.G.
- Suleiman, R.
- Tennant, C.
- Turner, D.L.
- Vidyaratne, L.S.

Kanto Information Service (KIS), Accelerator Group
Ibaraki, Japan
- Tajima, Y.

KEK
Ibaraki, Japan
- Aryshev, A.
- Enomoto, Y.
- Fujii, Y.
- Furukawa, K.
- Kaji, H.
- Kamikubota, N.
- Miyahara, F.
- Nakayoshi, K.
- Sakashita, K.
- Satake, I.
- Sato, M.
- Satoh, M.
- Sugimura, H.
- Terunuma, N.
- Urakawa, J.

KFE
Daejeon, Republic of Korea
- Lee, S.

KIP
Heidelberg, Germany
- Mkrtchyan, T.

KIT
Eggenstein-Leopoldshafen, Germany
- Beglarian, A.
- Blomley, E.
- Chilingaryan, S.A.
- Hoffmann, D.
- Kopmann, A.
- Mexner, W.
- Mostafa, J.
- Müller, A.-S.
- Schuh, M.

KOMAC, KAERI
Gyeongju, Republic of Korea
- Cho, S.Y.
- Dang, J.J.
- Kim, J.H.
- Lee, S.
- Song, Y.G.
- Yun, S.P.

Lancaster University
Lancaster, United Kingdom
- Gatignon, L.

LBNL
Berkeley, USA
- Barber, S.K.
- Felice, H.
- van Tilborg, J.

LIP
Lisboa, Portugal
- Cuim, F.M.O.
- Evans, G.G.
- Fernandez, R.P.
- Gomes, A.
- Gurriana, L.
- Martins, F.

LLNL
Livermore, California, USA
- Awwal, A.A.S.
- Barnes, A.I.
- Beaulac, L.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- Bhasker, A.
- Blackwell, B.
- Brunton, G.K.
- Budge, T.S.
- Burns, K.
- Casey, A.D.
- Castro Morales, J.R.
- Clark, R.D.
- Davis, B.A.
- Dixon, J.
- Estes, C.M.
- Fallejo, R.N.
- Fedorov, M.
- Fishler, B.T.
- Flegel, M.S.
- Gopalan, V.K.
- Hackel, B.
- Heerey, S.
- Koning, D.J.
- Koning, P.
- Lacuata, R.
- Leach, R.R.
- Lowe-Webb, R.R.
- Mathisen, D.G.
- McDonald, R.J.
- Miller Kamm, V.J.
- Muralidhar, S.
- Pacheu, V.
- Pan, Y.
- Patankar, S.
- Patel, B.P.
- Paul, M.
- Rozenshteyn, R.
- Sanchez, R.J.
- Sauter, S.
- Taranowski, M.
- Townsend, S.L.
- Tucker, D.
- Vaher, J.L.
- Van Wonterghem, B.M.
- Wang, J.
- Weaver, S.
- Wilhelmsen, K.C.
- Wilson, B.A.
- Zhang, H.

LNF-INFN
Frascati, Italy
- Buonomo, B.
- Di Giovenale, D.
- Di Giulio, C.
- Foggetta, L.G.
- Piermarini, G.
- Pioli, S.

LNLS
Campinas, Brazil
- Alnajjar, D.
- Aranha, P.D.
- Araujo, D.H.C.
- Arruda, L.C.
- Barreto, G.T.
- Bernardi, M.L.
- Brito Neto, J.L.
- Bueno, C.S.N.C.
- Calcanha, M.P.
- Camacho, L.U.
- Canova, H.F.
- Capovilla, L.G.
- Cardoso, F.H.
- Coelho, E.P.
- de Albuquerque, G.S.
- Del Nero, F.A.
- Do Carmo, L.P.
- Donatti, M.M.
- Franca, J.V.B.
- Franco, J.G.R.S.
- Furusato, F.S.
- Geraldes, R.R.
- Guedes, L.C.
- Horita, A.Y.
- Kofukuda, L.M.
- Kontogiorgos, G.N.
- Lena, F.R.
- Luiz, S.A.L.
- Martins dos Santos, L.
- Miqueles, E.X.
- Moraes, M.A.L.
- Moreno, G.B.Z.L.
- Nallin, P.H.
- Peixinho, A.Z.
- Perissinotto, L.S.
- Perna, A.V.
- Piccino Neto, A.C.
- Piton, J.R.
- Polli, R.W.
- Řežende, J.H.
- Rodrigues, G.L.M.P.
- Roque, C.S.B.N.
- Sanfelici, L.
- Saveri Silva, M.
- Segalla, L.F.
- Semissatto, G.T.
- Souza, M.S.
- Spina, T.V.
- Tolentino, H.C.N.
- Volpe, L.M.

Lund Institute of Technology (LTH), Lund University
Lund, Sweden
- Janneck, J.W.

MAX IV Laboratory, Lund University
Lund, Sweden
- Amjad, A.
- Aurelius, O.
- Barczyk, A.
- Bartalesi, A.
- Bell, P.J.
- Bertrand, B.
- Eguiraun, M.
- Enquist, H.
- Eriksson, T.
- Forsberg, J.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

- Freitas, Á.
- Gonzalez, A.
- Hardion, V.
- Jagudin, E.
- Klingberg, J.M.
- Li, Y.L.
- Lima, G.
- Lindberg, M.
- Lindhé, I.
- Matej, Z.
- Meirose, B.
- Nan, J.
- Nguyen, L.M.
- Padmanabhan, S.
- Petersson, J.E.
- Rosenqvist, J.T.K.
- Saad, M.
- Salnikov, A.
- Schurmann, J.
- Sjöblom, P.
- Skovhede, K.
- Spruce, D.P.
- Takahashi, C.

Max Planck Institute for the Physics of Complex Systems
Dresden, Germany
- Weninger, C.

MIPT
Dolgoprudniy, Moscow Region, Russia
- Nozik, A.A.

MLZ
Garching, Germany
- Brandl, G.

MPI/IPP
Garching, Germany
- Winter, A.

Nanjing University, College of Engineering and Applied Sciences
Nanjing, People's Republic of China
- Chu, C.P.

Naples University Federico II, Science and Technology Pole
Napoli, Italy
- Arpaia, P.

National Instruments Japan Corporation
MInato-ku, Tokyo, Japan
- Kamoshida, A.

NBI
København, Denmark
- Johnsen, C.
- Skovhede, K.
- Vinter, B.

NCBJ
Świerk/Otwock, Poland
- Bartoszek, P.R.
- Chmielewski, K.
- Kostrzewa, K.
- Kowalski, T.
- Markowski, P.
- Rybka, D.
- Sitek, M.
- Szewiński, J.
- Wojciechowski, Z.

NTUA
Athens, Greece
- Paraskevopoulos, C.
- Tzanis, P.

NVIDIA
Santa Clara, USA
- Kuhn, A.
- Nienhaus, M.

OCU
Osaka, Japan
- Hisano, A.
- Iwasaki, M.

ODU
Norfolk, Virginia, USA
- Iftekharuddin, K.M.
- Rahman, M.

ORNL
Oak Ridge, Tennessee, USA
- Hoffmann, C.M.
- Kasemir, K.-U.
- Vodopivec, K.
- White, K.S.
- Williams, D.C.

ORNL RAD
Oak Ridge, Tennessee, USA
- Calder, S.
- Mahoney, K.L.

Osaka University, Institute for Datability Science
Oasaka, Japan
- Nagahara, H.
- Nakashima, Y.
- Takemura, N.

PAL
Pohang, Republic of Korea
- Cho, W.S.
- Kim, S.W.
- Ku, K.H.
- Lee, W.W.

Pavia University
Pavia, Italy
- Fiorina, D.

PINSTECH
Islamabad, Pakistan
- Ajmal, M.
- Majid, A.
- Nawaz, D.A.
- Saqib, N.U.
- Sher, F.
- Tanvir, A.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PKU
Beijing, People's Republic of China
- Lin, C.

PNPI
Gatchina, Leningrad District, Russia
- Kazarov, A.G.

PSI
Villigen PSI, Switzerland
- Johansen, E.
- Kalantari, B.
- Koprek, W.
- Pollet, P.
- Theidel, G.
- Zimoch, D.

RadiaSoft LLC
Boulder, Colorado, USA
- Abell, D.T.
- Bruhwiler, D.L.
- Bruhwiler, K.
- Carlin, E.G.
- Coleman, S.J.
- Cook, N.M.
- Diaw, A.
- Edelen, J.P.
- Hall, C.C.
- Keilman, M.V.
- Moeller, P.
- Nagler, R.
- Nash, B.
- Pogorelov, I.V.
- Webb, S.D.

RBI
Zagreb, Croatia
- Cosic, D.D.
- Vićentijević, M.

RCNP
Osaka, Japan
- Nakano, T.

Reservoir Labs
New York, USA
- Langston, M.H.
- Lethin, R.
- Letourneau, P.D.
- Wei, J.

RIKEN Nishina Center
Wako, Japan
- Fujimaki, M.
- Fukunishi, N.
- Higurashi, Y.
- Ikezawa, E.
- Imao, H.
- Kamigaito, O.
- Kidera, M.
- Komiyama, M.
- Kumagai, K.
- Nagatomo, T.
- Nakagawa, T.
- Nishi, T.
- Ohnishi, J.
- Ozeki, K.
- Sakamoto, N.
- Suda, K.
- Uchiyama, A.
- Watanabe, T.
- Watanabe, Y.
- Yamada, K.

RIKEN/SPring-8
Hyogo, Japan
- Fukui, T.

ROE, UTAC
Edinburgh, United Kingdom
- Bridger, A.

S2Innovation
Kraków, Poland
- Celary, M.
- Gandor, M.
- Goryl, P.P.
- Kedron, K.
- Kowalski, G.W.
- Liszcz, M.
- Nabywaniec, M.
- Żytniak, Ł.

Sapienza University of Rome
Rome, Italy
- Ricci, G.

SARAO
Cape Town, South Africa
- Bode, W.A.
- Joubert, A.F.
- Madisa, K.
- Ockards, M.T.
- Swart, P.S.
- Twum, S.N.
- Venter, A.J.
- Wai, S.

SARI-CAS
Pudong, Shanghai, People's Republic of China
- Ning, D.J.

Shanghai Institute of Technical Physics, Chinese Academy of Sciences
Shanghai, People's Republic of China
- Ding, L.

SHI Accelerator Service Ltd.
Tokyo, Japan
- Hamanaka, M.
- Kaneko, K.
- Koyama, R.
- Nakamura, T.
- Ohki, T.O.
- Oyamada, K.
- Tamura, M.
- Yamauchi, H.
- Yusa, Y.A.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

SKAO
Macclesfield, United Kingdom
- Bartolini, M.
- Bridger, A.
- Miccolis, M.
- Rees, N.P.
- Santander-Vela, J.
- Venter, A.J.
- Vrcic, S.
- Yilmaz, U.

SLAC
Menlo Park, California, USA
- Edelen, A.L.
- Mayes, C.E.
- Roussel, R.J.
- Tao, F.

Sokendai
Ibaraki, Japan
- Popov, K.
- Wang, D.
- Yang, M.

SOLEIL
Gif-sur-Yvette, France
- Abeillé, G.
- Abiven, Y.-M.
- Akinotcho, S.
- Amelineau, L.
- Bourgoin, C.B.
- Briquez, F.
- Corruble, D.C.
- Da Silva, J.
- Elkaim, E.
- Engblom, C.
- Leluan, B.
- Lestrade, A.
- Monteiro, P.
- Munoz, L.E.
- Noureddine, A.
- Pilliaud, B.
- Pinty, V.
- Polack, F.
- Sebdaoui, M.
- Thibaux, G.
- Valléau, M.
- Zhang, S.

Soreq NRC
Yavne, Israel
- Isakov, H.
- Perry, A.
- Reinfeld, E.
- Shmuely, I.
- Solomon, Y.
- Tamim, N.

SSRF
Shanghai, People's Republic of China
- Bai, D.P.
- Chen, G.H.
- Chen, J.F.
- Ding, J.G.
- Li, M.
- Liu, X.M.
- Liu, Y.J.
- Lv, H.H.
- Mi, Q.R.
- Miao, H.F.
- Wang, H.Y.
- Yan, Y.B.
- Yu, C.L.
- Yu, P.X.
- Zhao, H.
- Zhao, Z.T.

STFC/DL
Daresbury, Warrington, Cheshire, United Kingdom
- Maheshwari, M.

STFC/DL/ASTeC
Daresbury, Warrington, Cheshire, United Kingdom
- Dunning, D.J.
- Gilfellon, A.J.
- Pollard, A.E.

STFC/RAL/ISIS
Chilton, Didcot, Oxon, United Kingdom
- Akeroyd, F.A.
- Aljamal, B.R.
- Baker, K.R.L.
- Baker, K.V.L.
- Burridge, R.A.
- Cole, L.
- Finch, I.D.
- Frank, W.A.
- Harper, J.R.
- Howells, G.D.
- Keymer, D.P.
- King, J.C.
- Lilley, S.
- Long, A.J.
- Löhnert, T.
- Malinowski, S.M.
- Medley, S.A.
- Moreton-Smith, C.
- Oram, D.E.
- Rai, B.
- Romanovschi, M.
- Saoulis, A.A.
- Washington, R.A.

Stockholm University
Stockholm, Sweden
- Bohm, C.
- Motzkau, H.
- Silverstein, S.B.

Tata Consultancy Services
Pune, India
- Kalsi, S.S.

Tbilisi State University
T'bilisi, Georgia
- Tsotskolauri, P.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-221-9 ISSN: 2226-0358

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Tessella
Abingdon, United Kingdom
- Jones, M.D.

TIFR
Colaba, Mumbai, India
- Bheesette, S.
- Elangovan, Y.
- Majumder, G.
- Panyam, N.

Tsinghua University
Beijing, People's Republic of China
- Gong, G.H.

TUB
Beijing, People's Republic of China
- Ye, Y.M.

TUL-DMCS
Łódź, Poland
- Makowski, D.R.
- Mazur, P.
- Perek, P.

UCT Physics
Cape Town, South Africa
- Lunt, S.A.

UNICAMP
Campinas, São Paulo, Brazil
- Silva, G.G.

Universidad de La Frontera
Temuco, Chile
- Augsburger, R.A.
- Carrasco, S.A.
- Galeas, P.
- Huenupan, F.
- Seguel, R.S.

Universidade do Porto, Faculdade de Ciências
Porto, Portugal
- Ribeiro, H.R.

University of Cape Town
Cape Town, South Africa
- Teixeira, D.I.

University of Chicago
Chicago, Illinois, USA
- Gonzalez-Aguilera, J.P.
- Kim, Y.K.

University of Malta, Information and Communication Technology
Msida, Malta
- Valentino, G.

University of Naples Federico II
Naples, Italy
- Gargiulo, F.

University of Pittsburgh
Pittsburgh, Pennsylvania, USA
- Hong, T.M.
- Roche, S.T.

UPM-I2A2
Madrid, Spain
- Astrain, M.
- Costa, V.
- Rivilla, D.
- Ruiz, M.

USTC
Hefei, Anhui, People's Republic of China
- Gu, G.
- Jiang, Z.Y.
- Yin, J.
- Zhao, L.

USTC/NSRL
Hefei, Anhui, People's Republic of China
- Huang, Z.Y.
- Li, C.
- Li, W.
- Liu, G.
- Sun, X.K.
- Wang, J.G.
- Xu, S.
- Xu, W.
- Xuan, K.
- Yu, Y.B.
- Zhang, Z.