ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB259

HIGH LEVEL APPLICATIONS FOR SIRIUS ACCELERATORS CONTROL

A. C. de S. Oliveira*, M. B. Alves, F. H. de Sá, L. Liu, X. R. Resende
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

Sirius is a 4th generation 3 GeV synchrotron light source that has just finalised the first commissioning phase at the Brazilian Center for Research in Energy and Materials (CNPEM) campus in Campinas, Brazil. The large number of process variables and large complexity of the subsystems in this type of machine requires the development of tools to simplify the commissioning and operation of the accelerators. This paper describes some of the high level control tools developed for the accelerators commissioning and future operation.

INTRODUCTION

Sirius is the new 4th generation synchrotron light source built by the Brazilian Synchrotron Light Laboratory (LNLS) in the Brazilian Center for Research in Energy and Materials (CNPEM), in Campinas, Brazil. The machine consists of a 150 MeV Linac, built by SINAP [1], a full-energy Booster and a 3 GeV storage ring, designed with a multi-bend-achromat approach to reach the emittance of 250 pm rad [2].

The first phase of Sirius commissioning has just finalized and its first beamlines are coming into operation [3]. Some experiments by internal and external users were already registered. Since mid 2020, the machine has been operating in decay mode, with two injections per day, for beamline commissioning. In January 2021, there was a girders alignment campaign in all accelerators, which required a new commissioning in February. Since early April 2021, it has been operated with an initial current of 70 mA.

Some of the activities scheduled for this year are the first tests with top-up operation, fast orbit feedback (FOFB) system and orbit interlock implementation and some new beamlines installation.

In this article, we will present the high-level control tools that have been developed and used for machine commissioning and future operation. The following sections will bring an overview of the control system (CS) and its support applications used and describe the high-level applications (HLA) available, such as: services to record and consult the machine's history and state, as well as CS constants information; the generic analysis applications most used for experiments and diagnostics; and, finally, specialized applications for monitoring and controlling subsystems and physical parameters.

CONTROL SYSTEM OVERVIEW

Sirius accelerators are composed of several subsystems, most of them controlled through an *EPICS* [4] communication architecture, more specifically the version R3.15.

The control of accelerator devices involves using around 500 device and soft input/output controllers (IOC), which run in 10 servers monitored with the help of *Zabbix* [5] service. Besides others support applications, these IOCs are distributed in 113 docker stacks and 325 containers, which are managed with the *Portainer* [6] tool.

Through these IOC, a wide variety of machine parameters are configured and monitored through around 150 thousands of process variables (PV), number that is expected to increase after the integration of some subsystems to the architecture and the insertion of new devices to the machine. In order to control all these PVs in the most intuitive and effective way, so as to complete commissioning tasks and achieve design parameters, a wide variety of HLA has been developed in-house in a collaborative way, mostly using the *GitHub* [7] and *GitLab* source version control platforms, with open-source tools. In the EPICS community, there are many choices for high-level application development tools, which have been well deployed on Sirius, both on the EPICS server [8] and client sides.

Configuration of control room desktop computers, service docker image and containers are managed with *Ansible* automation tool [9]. The open-source package management tool *Conda* [10] has also been applied to manage environment and dependencies of HLA.

RECORDING AND SHARING SERVICES

We have been using an instance of the *EPICS Archiver Appliance* [11] to record the PVs history. Besides the standard archiver maintenance tool, another PV management tool has been developed in-house to simplify and speed up the handling of large volumes of PVs.

EPICS Archiver has a Viewer application, also developed in-house [12]. It is a web interface, implemented with *TypeScript* [13], using *JavaScript* libraries such as *React* [14] and *Redux* [15]. The viewer, which is shown in Fig. 1, is a tool used daily for monitoring and faults diagnosis, which has been quite efficient for history visualization, with very useful features such as zoom, scale and limit adjustments.

Figure 1: Snapshot of the in-house developed Archiver Viewer.

* ana.clara@lnls.br

12th Int. Particle Acc. Conf. IPAC2021, Campinas, SP, Brazil JACoW Publishing
ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB259

The in-house *python* packages developed are now well integrated to recover Archiver data. These packages have been improved throughout the development of analysis tools and machine performance reports.

To register the machine states and also to retrieve known configurations, we have the in-house developed configuration service, called *ConfigDB*. The service consists of a *Flask* [16] server, with REST protocol, which manages a *mongoDB* [17] database, and has a *python* API, which uses the *PyMongo* [18] library. On the client side, we also have a small *python* layer. It was a service developed to provide flexibility about the type of the data that can be registered and easy to extend. New types of configuration can be defined at any time, which can be available to be used after a task as simple as defining a template in a *python* file. The two most commonly used types of configuration are machine snapshots and correction parameter configurations, such as orbit response matrices.

The service is accompanied by graphical user interfaces (GUI), based on *PyQt5* [19], for consulting configurations data, saving and applying new snapshots. For those, the interfaces use the *PyEpics* [20] library. Figure 2 shows a GUI for applying a configuration. It has a PV selection tool, with a tree view with division logic based on the Sirius naming system [2, 21], which make it easy to apply a partial state.

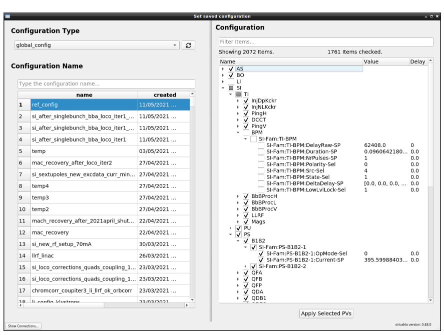

Figure 2: *ConfigDB - Apply* GUI.

To ensure the synchronization of CS constants information in all applications, such as magnet excitation curves, device positioning and low level communication architecture details, we have a static table server available via http Apache [22]. For each table, there is a simple *python* layer for convenient availability of the data, developed in order to allow multiple simultaneous queries to the data and to minimize the queries to the server. The service is used both by device (power supplies) and soft IOCs (high level timing), as well as by GUI implemented to control these subsystems.

For recording and sharing machine study data, the control room desktops are connected in a network scalable filesystem structure. We recently migrated this sharing scheme from NFS to *GlusterFS* [23].

ANALYSIS AND DIAGNOSTIC TOOLS

A variety of generic tools for visualization and analysis are available in the control room desktops. The most used tool for PV history visualization and monitoring is the Archiver Viewer, presented in the previous section.

Another visualization tool used in operation activities is the software *Igor Pro* [24], an initiative by WaveMetrics, that runs in Windows. The software was widely used in the UVX, the first synchrotron light source built by LNLS, and was adapted to communicate with the Archiver Appliance. It is a very interesting tool that brings options such as easy macro development, several markers, with options to check differences between selected points. Currently, the LNLS operation group has been working to provide more options for data visualization, such as history comparisons of parameters at different time intervals.

An alternative and simple tool that we keep available on the control room desktops is the *TimeChart* [25] viewer, an initiative by the same developers from *PyDM* [26] project. It allows online time tracking of several PVs and it is interesting for scanning experiments.

The tool that has been most used for machine studies is, undoubtedly, *Jupyter Notebook* [27]. Python notebooks with scripts and generic analysis, such as response matrices and betatron function measurements, are organized in the file sharing service and replicated for the various experiments. The *python* environment, with libraries like *Numpy* [28] and *SciPy* [29], for data processing, and *Matplotlib* [30], for data visualization, has proven to be very useful, versatile and robust for several types of analysis. The development of tools to integrate the characterization study scripts with the CS has been extensively explored by the accelerator physics group. Throughout the Sirius project, the group worked on the implementation of simulation libraries, in *c++* and *python* languages. Recently, a great effort has been made to expand and improve these tools to integrate them with the CS. This integration, of simulation and CS interaction environments in the same language, greatly facilitates the development of new tools and has made it possible to implement more complex analysis, such as the LOCO algorithm [31], for the machine linear optics studies, and beam position monitors (BPM) turn-by-turn (TbT) data analysis, for estimates of equilibrium parameters and nonlinear optics [32].

As secondary tools, *Matlab* [33] and *Octave* [34] are also available on the control desktops, as well as the *AT* [35] simulation environment, prepared for EPICS communication through *LabCA* [36] and *MCA* [37] libraries.

MONITORING AND CONTROL TOOLS

The vast majority of desktop applications used in the machine operation are developed in *PyQt5*, *PyEpics* and *PyDM*. These libraries have proven to be very versatile, complete and practical for creating applications that provide to the user simultaneous control and monitoring of various devices, flexibility in selecting devices and commands, triggering synchronized command sequences. Magnets cycling interface,

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB259

power supply management (shown in Fig. 3), tune and orbit monitoring and Booster ramp control are some of the examples of applications that use these libraries.

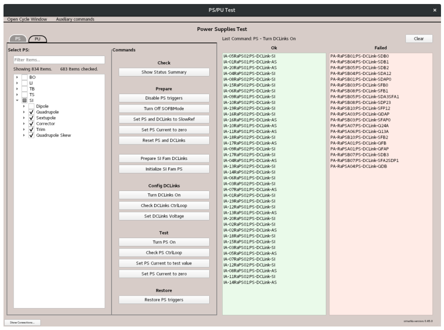

Figure 3: *PSTest*, a power supply management application.

The HLA development is guided by maintaining in soft IOCs, mostly implemented using the *PCASpy* [38] and *PyEpics* libraries, the parameters that define machine states and that must be accessible by any EPICS client, for control or archiving tasks. Control windows combine minimal control logic, command sequences and data visualization tools that make control more intuitive and convenient. For this reason, the most used applications in daily controls are formed by these two layers, as is the case of applications such as Trajectory and Orbit Correction and Analysis (TOCA), whose GUI is illustrated by Fig. 4, power supply monitoring and diagnosis, injection efficiency calculation, beam lifetime computation, tune correction, among other applications.

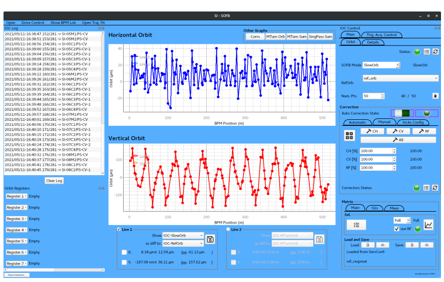

Figure 4: The TOCA GUI.

Some more specific and low level system controls are still available on interfaces developed with the *CS Studio* [39] tool, such as radio frequency, timing and beam diagnostics. For Linac, most of the controls are still available on *EDM* [40] interfaces, provided by SINAP, and part of them has already been migrated to *PyDM*.

Although the control room desktops have been planned to be based on Linux and most HLA have been designed and developed for this operational system, efforts on multi-platform development have grown to facilitate access to machine status and CS data outside the control network. This is a demand that has been growing, both for access by other groups in CNPEM campus and for remote access by people who are in home office due to the current pandemic status. One of the strategies adopted to meet these demands is the development of web applications. Figure 5 shows the storage ring vacuum pressure monitor, an example of a web application, implemented in *JavaScript* with *React* library and that uses *epics2web* package [41,42], an initiative from Jefferson Lab [43], that allows access to PVs via http requests or web sockets.

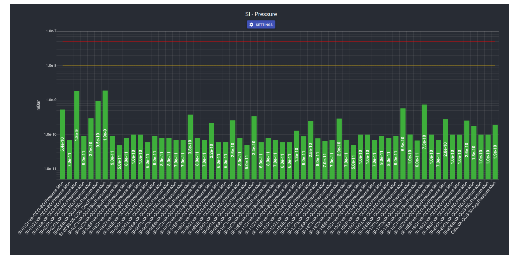

Figure 5: Storage Ring Vacuum Pressure Monitor.

Efforts to develop HLA are still foreseen, which includes the implementation of tools for: top-up operation; interlock system monitoring; PV correlation analysis; FOFB control and data analysis; insertion devices control; report automation and users beam quality analysis. In addition, it is also intended to work on tools which will be specialized to support operation tasks, which may be equipped with features like alarms, reports and audio alerts.

CONCLUSION

Sirius is a 4th generation synchrotron light source built by LNLS, whose control system is based on EPICS. Recently, its first commissioning phase was completed, an achievement that relied on the development of several support applications and high-level control tools.

Python, as well as its libraries and *Jupyter Notebook* environment, has been widely used to implement tools for daily operation and machine studies. The most used libraries for the development of desktop applications are *PyQt5*, *PyDM*, *PyEpics* and *PCASpy*. Great effort has been made in the integration of control and simulation libraries in this language, which has greatly facilitated the development of new tools and enabled the implementation of more complex analysis such as the LOCO algorithm and BPM TbT data analysis. *Javascript* libraries, such as *React* and *Redux*, along with the *epics2web* package, have also been widely used in the development of support applications and monitoring machine parameters GUI.

We intend to continue working on the integration of the CS applications, as well as on the improvement of the operation tools, to simplify tasks like fault diagnosis and investigation of correlation between system parameters.

ACKNOWLEDGEMENTS

The authors would like to thank Eduardo Coelho and Claudio Carneiro, from software and automation group, for providing information on CS architecture and management tools, besides on web applications under development; and the operation group, especially Rone Castro, on the developments for the *Igor Pro* tool.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB259

REFERENCES

[1] SINAP - Shanghai Institute of Applied Physics, Chinese Academy of Sciences, http://english.sinap.cas.cn/.

[2] The Sirius Project, https://wiki-sirius.lnls.br/.

[3] L. Liu, M. B. Alves, F. H. de Sá, A. C. S. Oliveira, and X. R. Resende, “Sirius Commissioning Results and Operation Status”, presented at the 12th Int. Particle Accelerator Conf. (IPAC’21), Campinas, Brazil, May 2021, paper MOXA03, this conference.

[4] EPICS - Experimental Physics and Industrial control System, https://epics.anl.gov/.

[5] Zabbix, https://www.zabbix.com/.

[6] Portainer, https://www.portainer.io/.

[7] Brazilian Synchrotron Light Laboratory Organization in Github, https://github.com/lnls-sirius

[8] X. R. Resende, L. Liu, A. C. S. Oliveira, F. H. de Sá, and G. L. do Prado, “High-Level Applications for the Sirius Accelerator Control System”, in *Proc. 10th Int. Particle Accelerator Conf. (IPAC’19)*, Melbourne, Australia, May 2019, pp. 2462–2465. doi:10.18429/JACoW-IPAC2019-WEPGW003

[9] Ansible, https://www.ansible.com/.

[10] Conda, https://docs.conda.io/en/latest/.

[11] The EPICS Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs/.

[12] EPICS Archiver Web Interface, https://github.com/lnls-sirius/archiver-viewer

[13] TypeScript - Typed JavaScript at Any Scale, https://www.typescriptlang.org/.

[14] React - A JavaScript library for building user interfaces, https://reactjs.org/.

[15] Redux - A Predictable State Container for JS Apps, https://redux.js.org/.

[16] Flask, https://palletsprojects.com/p/flask/.

[17] MongoDB - The database for modern applications, https://www.mongodb.com/.

[18] PyMongo, https://pymongo.readthedocs.io/en/stable/.

[19] PyQt5, https://pypi.org/project/PyQt5/.

[20] PyEpics: Epics Channel Access for Python, https://pyepics.github.io/pyepics/.

[21] K. Rathsman, G. Trahern, J. Malovrh Rebec, M. Rescic, and M. Vitorovic, “ESS Naming Convention”, in *Proc. 4th Int. Particle Accelerator Conf. (IPAC’13)*, Shanghai, China, May 2013, paper THPEA038, pp. 3222–3224.

[22] The Apache Software Foundation, https://www.apache.org/.

[23] Gluster, https://www.gluster.org/.

[24] Igor Pro, https://www.wavemetrics.com/.

[25] TimeChart, https://slaclab.github.io/timechart/index.html

[26] PyDM - Python Display Manager, https://slaclab.github.io/pydm/.

[27] JupyterLab, https://jupyter.org/.

[28] NumPy, https://numpy.org/.

[29] SciPy.org, https://www.scipy.org/.

[30] Matplotlib, https://matplotlib.org/.

[31] M. B. Alves, “Optics Corrections with LOCO on Sirius Storage Ring”, presented at the 12th Int. Particle Accelerator Conf. (IPAC’21), Campinas, Brazil, May 2021, paper MOPAB260, this conference.

[32] X. R. Resende, M. B. Alves, F. H. de Sá, and L. Liu, “Equilibrium and Non-Linear Beam Dynamics Parameters From Sirius Turn-by-Turn BPM Data”, presented at the 12th Int. Particle Accelerator Conf. (IPAC’21), Campinas, Brazil, May 2021, paper TUPAB219, this conference.

[33] MATLAB, https://www.mathworks.com/products/matlab.html

[34] GNU Octave, https://www.gnu.org/software/octave/index

[35] Accelerator Toolbox Collaboration, http://atcollab.sourceforge.net/.

[36] LabCA - EPICS/Channel Access Interface for Scilab and Matlab, http://www.slac.stanford.edu/~strauman/labca/index.html

[37] A. Terebilo, “Channel Access Client Toolbox for MATLAB”, in *Proc. 8th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’01)*, San Jose, CA, USA, Nov. 2001, paper THAP030, pp. 543–544.

[38] PCASpy, https://pcaspy.readthedocs.io/en/latest/.

[39] CS | Studio, https://controlsystemstudio.org/.

[40] EDM: Extensible Display Manager, https://www.slac.stanford.edu/grp/cd/soft/epics/extensions/edm/edm.html

[41] Epics2web, https://github.com/JeffersonLab/epics2web

[42] R. J. Slominski and T. L. Larrieu, “Web Extensible Display Manager”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’17)*, Barcelona, Spain, Oct. 2017, pp. 852–856. doi:10.18429/JACoW-ICALEPCS2017-TUPHA181

[43] Jefferson Lab, https://www.jlab.org/.