ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

DATA ACQUISITION SOFTWARE PIPELINE FOR THE COMMISSIONING OF THE LoKI SMALL ANGLE NEUTRON SCATTERING INSTRUMENT

J. Walker*, S. Alcock, M.J. Christensen, J.E. Houston, K. Muric, W. Potrzebowski, T.S. Richter
European Spallation Source ERIC, Lund, Sweden
Davide Raspino, UKRI-STFC, ISIS Neutron and Muon Source,
Harwell Science and Innovation Campus, Chilton, Oxfordshire, UK

Abstract

The LoKI Small-Angle Neutron Scattering (SANS) instrument will be one of the first instruments to be commissioned at the European Spallation Source (ESS), and will contribute to the early science programme. The detector for the instrument was tested at the ISIS neutron source facility in March of 2022, and this paper outlines the data acquisition software pipeline. It consists of a readout master, an Event Formation Unit (EFU), an instance of Kafka, a NeXus file writer, and, the data reduction software, Scipp. The readout master is responsible for synchronising detector readout timestamps with an external reference, and aggregating those readouts into UDP packets sent to the EFU. The EFU processes the readout data to produce event messages containing a location and time of arrival for each neutron event detected, and acts as a Kafka producer sending event messages to Kafka. The NeXus file writer consists of a Kafka consumer that compiles event messages and other experiment data from a given time interval into a single NeXus file for further analysis. Finally, Scipp is a data reduction python library used to visualise and analyse the experiment data after an experiment is completed.

INTRODUCTION

The European Spallation Source (ESS), co-hosted by Sweden and Denmark, is currently under construction in Lund (Sweden), with the first neutrons expected in 2024. A number of detectors are being developed for the facility, one of which is for the LoKI Small-Angle Neutron Scattering instrument [1, 2], which is being developed in colaboration with Science and Technology Facilities Council (STFC) in the U.K. LoKI will make use of ^{10}Boron-Coated Straws (BCS) from Proportional Technologies Inc. (USA). In order to obtain acceptable efficiencies, these detectors consist of 7 boron-coated copper straws packed within 1 inch aluminium tubes, with each copper straw wired as a position sensitive detector. On LoKI, these 1-shell-escape inch aluminium detector tubes will then be packed into arrays to make detector panels, which will be placed in 4-panel banks around the beam at ~1.3 m and ~4 m from the sample, and a single panel bank on a carriage, which will move between ~5 m and 10 m from the sample positions. Given the number of detector straws in this design, in order to minimise detector signal cables, the multiplexing method will be used. The multiplexing method consists of a resistive chain between the 7 straws at each end of the aluminium tube, resulting in four signals from each each tube rather than 14. When a neutron is detected, 4 amplitude values are then produced, from which the location of the neutron event can be calculated. A smaller subset of this detector consisting of 128 tubes was used for the tests outlined in this paper. A diagram of this detector is shown in Fig. 1, showing the layout of the tubes, and the layout of the straws within each of those tubes is shown in Fig. 2.

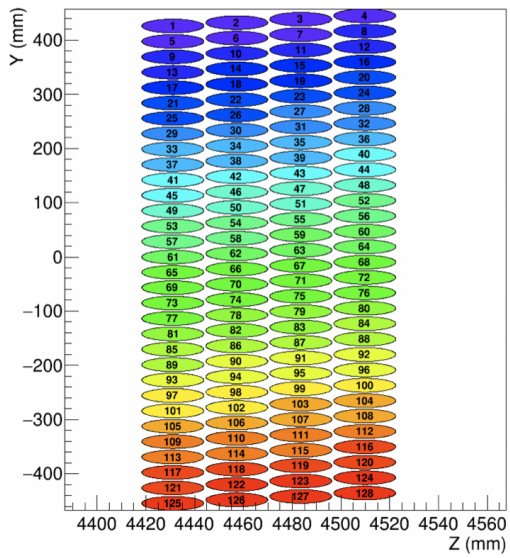

Figure 1: A diagram depicting the tubes of the subset of the LoKI detector tested at ISIS in March 2022.

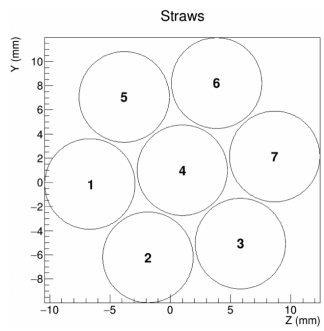

Figure 2: A diagram depicting the straws within each tube of the LoKI detector.

The detector was tested on the LARMOR instrument at the ISIS facility with the full data acquisition pipeline [3]. This pipeline will be used in production at the ESS facility. It consists of the following key components, developed by

* jennifer.walker@ess.eu

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

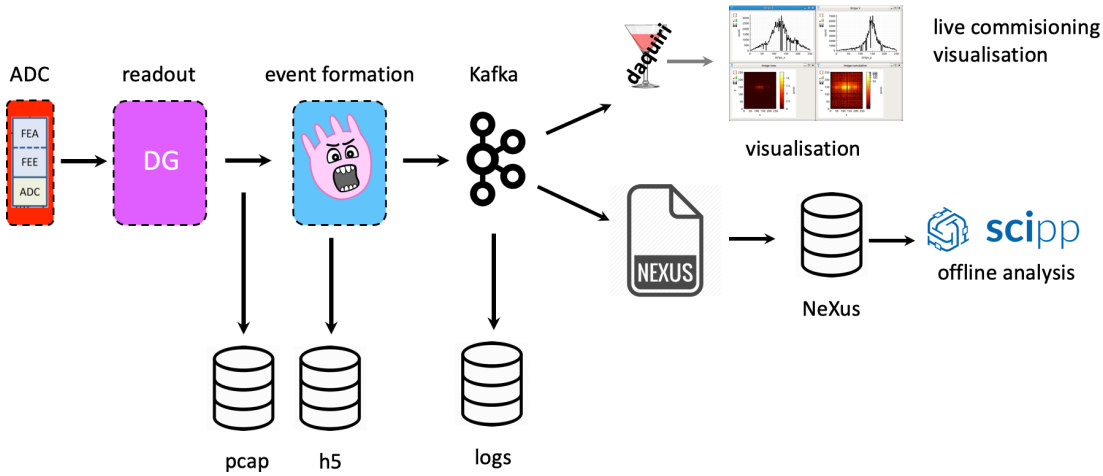

Figure 3: The data acquisition pipeline used at the LoKI test at ISIS in March 2022.

ESS and in-kind partners, each of which will be outlined in this paper.

- Readout Master
- Event Formation Unit
- NeXus File Writing
- Scipp Data Reduction

The data acquisition software chain is shown in Fig. 3, from the Analogue-to-Digital Converter (ADC) readouts on the detector, through the Readout Master [4], to the Event Formation Unit [5], where unprocessed data for this test is also stored in the form of pcap and h5 files for later analysis and reprocessing. The pipeline then goes on to the Kafka message broker[6], and services that consume data from Kafka in this case include DaqLite [7] and the NeXus file writer. DaqLite is a live data visualisation tool that allows for monitoring of the detector activity early in the acquisition chain. It displays the cumulative hit count at each voxel of the detector. The files written by the NeXus file writer are then ready for data reduction in Scipp [8, 9].

READOUT MASTER

Detector systems convert neutron events into electrical charges, which can be amplified, shaped, and digitised. Large detector systems may comprise many thousands of such outputs. The digitisation may be performed by a series of discrete ADCs, or by more sophisticated multi-channel ASICs. These devices must be read out, typically by one or many FPGAs. The function of the ESS Readout Master is to simplify the communication between these FPGAs and the relevant downstream systems provided by the Integrated Controls Group (ICS) - slow control and timing - and the Data Management and Software Centre (DMSC) - event formation. The Readout Master essentially acts as a point of aggregation, allowing for a common, standardised interface between the detector readout and the ICS/DMSC.

For the ICS interface, the Readout Master must handle slow control and timing distribution. For the timing interface, the Readout Master instantiates the open-source MRF (Micro Research Finland) EVR (Event Receiver) firmware, allowing it to connect to the ESS facility MRF timing system. This allows for the recovery of the ESS facility clock, timestamp and other useful events distributed by the MRF system, for example those related to the accelerator pulse. The clock and timestamp are forwarded to so-called "Front End Nodes" (FENs) which interface to the front end FPGA/ADC/detector chain. The connectivity between the Readout Master and the FENs is achieved through 8B/10B encoded MGT-driven (Multi-Gigabit Transceiver) optical fibres links arranged in a ring topology. Up to 12 rings of 31 FENs are supported, allowing for a scalable system from 1 to 372 FENs, with each FEN typically supporting a minimum of 64 ADC channels. Each FEN can recover the ESS clock from the MGT links. The timestamps are synchronised for each FEN using a simple round-trip delay calculation. Timestamp distribution is accurate to within 100 nanoseconds, and is guaranteed not to drift with respect to the ESS facility timestamp.

For the slow control interface, the Readout Master instantiates a light-weight UDP stack, which communicates with the EPICS IOC (Input Output Controller) using a standard Gigabit Ethernet link. The slow control system is used to read and write to all registers within an instrument readout system. The EPICS IOC acts as the slow control Master, and the Readout Master is responsible for routing register requests to the relevant address space. If the required register is on a FEN, the request and response are handled by the same optical links which carry the timing information.

The FENs package neutron event data and send it back to the Readout Master over the same optical links that carry the timing information. The Readout Master aggregates these into UDP packets, appends relevant metadata, and then forwards them to the EFU (Event Formation Unit) as jumbo frames over one or two 100 Gigabit Ethernet links [10].

EVENT FORMATION UNIT

The Event Formation Unit, a Linux server application written in C++, receives data from the Readout Master from the previously described 100 Gigabit Ethernet links. The data format output by the Readout Master for consumption

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

by the Event Formation Unit consists of the ESS data header, which among other things defines the pulse time and previous pulse time, and a series of LoKI readouts. The format is shown in Fig. 4. Each of these readouts represents a single neutron event, and all readouts in a single packet belong to the same pulse time, indicated in the ESS header.

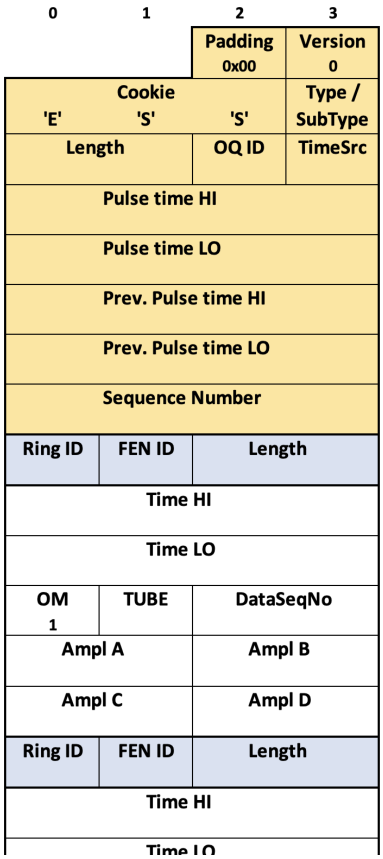

Figure 4: The ESS Header data format, in yellow, followed by a series of LoKI readouts. The format is displayed as 4 bytes wide and offset by 2 bytes in the diagram for legibility.

The Event Formation Unit is responsible for processing these readouts and outputting Kafka messages containing their pulse time, time of flight, and pixel ID. The pixel ID is an integer value that represents the location on the detector where each neutron is detected. The definition of these pixel IDs is referred to as the Logical Geometry of the detector, and is shown in Fig. 5. Mapping from the digital identifiers in a readout - the Ring ID, FEN ID, and Tube number - is defined in the Digital Mappings, shown in Fig. 6.

The straw number and the location along each straw are calculated from the four amplitude values per readout according to the following equations:

$$LocalStraw = \frac{B+D}{A+B+C+D}6$$

$$Position = \frac{A+B}{A+B+C+D}(2^N - 1)$$

With the resolution of position, the value N, being 512 for these tests.

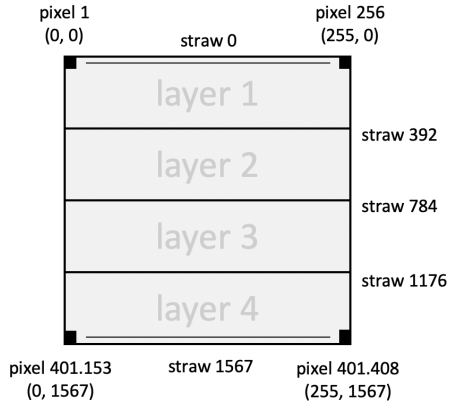

Figure 5: The Logical Geometry, defining an integer value per position on the detector.

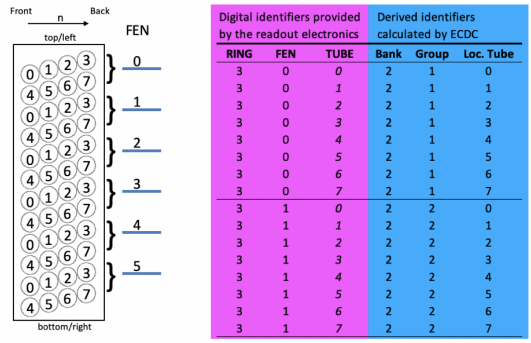

Figure 6: The Digital Mapping, defining what Ring ID, FEN ID, and Tube numbers map to specific tubes in the detector.

From the local straw value, i.e. the straw number within a tube, and the global tube position, a global straw number is calculated. This then allows the final pixel ID calculation:

$$PixelID = 1 + Position + GlobalStraw * 2^N$$

Finally, the Event Formation Unit produces Kafka messages using the ev42 flatbuffers schema consisting of the following fields:

Field Name	Type	Description
`source_name`	`string`	Field identifying producer, commonly detector name
`message_id`	`ulong`	Consecutive numbers, used to check order of messages
`pulse_time`	`ulong`	Nanoseconds since Unix epoch of given pulse time
`time_of_flight`	`uint`	Nanoseconds since pulse for given event
`detector_id`	`uint`	Pixel ID, identifying where neutron was detected

NeXus FILE WRITING

Data file writing is performed using an in-house developed file writer, called Kafka-to-nexus, written in C++. As the name indicates, it retrieves all the collected data during the experiment from a Kafka cluster and writes it to a NeXus

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

```
{
    "module":"ev42",
    "config":
        {
            "topic":"loki_detector",
            "source":"loki"
        }
}
```

Figure 7: The LoKI event data configuration for the file writer. As can be seen, the ev42 flat buffer schema is used to serialise the event data. The file writer can fetch the collected data from the loki_detector Kafka topic.

file. NeXus is a format based on HDF5 that is commonly used in muon, neutron and x-ray science. Attributes are used extensively; for example, group attributes are used to define the NeXus class of a group written to the file.

At ESS, the overall structure of the NeXus file and the type of data written is defined by a template JSON file, which is provided to the file writer in a start job message sent to a file writer command Kafka topic. In addition to where and how to fetch event and measurement data, the template JSON file also contains static data, which can be written directly by the file writer from the NeXus template structure. One such instance of static data would be any beforehand known experiment information, such as experiment title or beam users. Any dynamic data that needs to be written in the NeXus file is described by the template file where such a stream data configuration, for event data using the ev42 schema, is shown in Fig. 7.

The pixel geometry information and ID mapping is also provided through the NeXus template file. As the shapes are uniform for each pixel in the LoKI detector, the NXcylindrical_geometry class according to the NeXus standard is used to describe a pixel. Data sets x_offsets, y_offsets and z_offsets are used to provide each individual pixel with Cartesian coordinates with the origin at the sample position. Transformations according to the NXtransformation class are used to place the detector correctly in the global coordinate system relative to the sample (origin). Both the data sets and the pixel shape groups are items in an NXdetector group. Additionally, in the NXdetector group there is a data set called detector_number that defines the mapping of pixels to pixel IDs. This mapping is defined by the Logical Geometry of the detector in combination with physical dimensions specified in technical specifications, such as CAD drawings. Event data with the configuration according to the one in Fig. 7 is written to an NXevent_data class, which is a subgroup of the NXdetector group.

A start job command containing the start time of the write job and the previously described JSON template file is sent to the file writer command topic. A file writer is selected from a pool of currently idle and available writers, and it will start writing the data based on the provided template. First of all, the NeXus file writer defines the overall structure of the file and populates it with the static data such as pixel geometries, experiment information. Secondly, it will start writing all dynamic measurement and event data based on provided flatbuffer schema, Kafka topics and sources.

The data writing will either stop after a preset amount of time or after the file writer receives an explicit stop job message. Once the file writer stops writing the dynamic data, it will proceed to generate some additional metadata, such as average, minimum, and maximum values of data sets written to the NeXus file. Once all this is done, the file writer will finally emit a writing done message to Kafka to notify that it is done writing a NeXus file and switch its state from busy to idle. An example of a file written in the detector tests is displayed in Fig. 8.

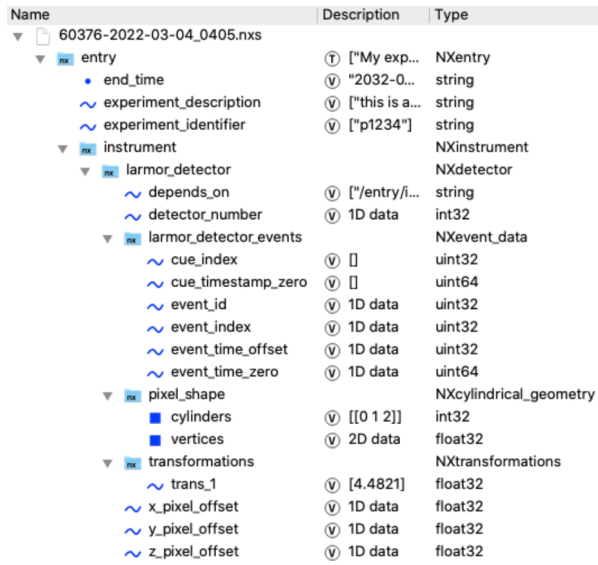

Figure 8: NeXus file with the structure written in the LoKI detector tests.

DATA REDUCTION WITH Scipp

Scipp is data reduction framework developed at ESS [8, 9]. It features handling of physical units, propagation of uncertainties, optimised data structure for multidimensional arrays, support for binned and event data, instrument visualisation, flexible masking and plotting. In order to provide flexibility, easy development and testing an overall scipp ecosystem consists of four main modules *scipp*, *scippnexus*, *scippneutron* and *ess* (Fig. 9). *scipp* is general package utilising multi-dimensional arrays of data with named dimensions and associated coordinates. *scippnexus* provides a link between the HDF5-based NeXus Data Format and *scipp*. *scippneutron* is specifically designed for handling neutron scattering data reduction by providing "unit conversions" and technique specific tools based on physics of (time-of-flight) neutron scattering. *ess* provides ESS facility and instrument bespoke tools for neutron data reduction and visualisation. All *scipp* modules are easily accessible from Jupyter-notebooks (*ess-notebooks*), which contain both code and documentation.

During LoKI detector test a few notebooks (at *ess-notebooks* level) were created primarily for debugging and

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

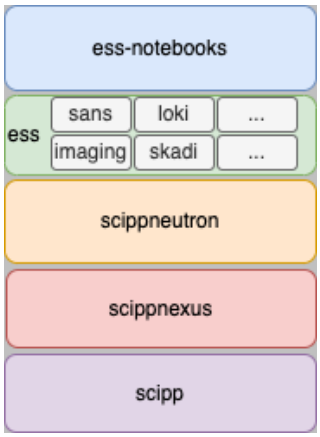

Figure 9: Scipp ecosystem architecture.

sanity checks. The main functionality of the notebooks involved: loading files, printing basic metrics (e.g number of recorded events), showing instrument view (Fig. 10) and plotting TOF spectra of detector and monitors. By performing such tests we were able to quickly identify issues with incorrect pixel positions and missing data.

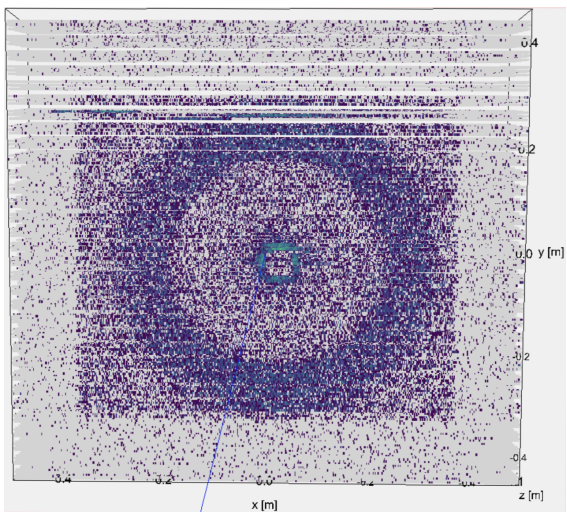

Figure 10: Silver behenate SANS data viewed with scipp's instrument view.

Once measurements were finished, the recorded data was further processed by performing series of calibrations and data reduction. The calibration data was collected by placing masked strips in front of the detectors with a uniformly scattering sample. The obtained peaks positions were used as a reference for further corrections and fitted using Mantid software [11] (it is anticipated that in future such corrections will be also performed in *scipp*). Corrected files were subsequently passed to *scipp* where data sets from different runs were merged. The merged files were subject of further data reduction protocol as demonstrated in (https://scipp.github.io/ess/techniques/sans/sans.html). In brief the protocol involves loading files, defining masking, applying coordinate corrections and then converting from TOF to lambda space and from lambda to Q space. Eventually background data is subtracted from sample and normalised (Fig. 11). The final outcome of data reduction protocol is intensity curve in the function of scattering vector, q.

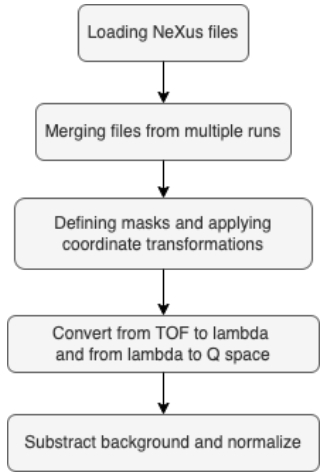

Figure 11: Data reduction workflow.

CONCLUSION

The LoKI test at ISIS in March 2022 demonstrated the functionality of the ESS data acquisition pipeline from the detector electronics, through FPGAs, into software, and into a state that is easily analysed and used for science. Furthermore, it demonstrated the ability to support the experiment remotely through remote diagnostics and debugging. The Readout Master aggregates detector readouts into packets, which are sent to the EFU to be processed into event data. The event data is sent to Kafka, where it is then used by the NeXus Filewriter to write a NeXus file per experiment, which is then analysed using scipp. This entire pipeline was running and live experiment data was processed in much the same way that it will be in production at the ESS facility.

REFERENCES

[1] K. Andersen *et al.*, "The instrument suite of the European Spallation Source," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 957, p. 163 402, 2020. doi:10.1016/j.nima.2020.163402

[2] G. Náfrádi *et al.*, "Shielding considerations for ESS LoKI," *J. Neutron Res.*, vol. 22, no. 2-3, pp. 119–129, 2020. doi:10.3233/JNR-200152

[3] A. Mukai *et al.*, "Architecture of the data aggregation and streaming system for the European Spallation Source neutron instrument suite," *J. Instrum.*, vol. 13, no. 10, T10001, 2018. doi:10.1088/1748-0221/13/10/t10001

[4] S. Kolya *et al.*, "Building a research infrastructure and synergies for highest scientific impact on ESS, Deliverable 4.1 – Integration Plan for Detector Readout," 2015. doi:10.17199/BRIGHTNESS.D4.1

[5] M. Christensen *et al.*, "Software-based data acquisition and processing for neutron detectors at European Spallation Source—early experience from four detector designs," *J. Instrum.*, vol. 13, no. 11, T11002, 2018. doi:10.1088/1748-0221/13/11/t11002

ISBN: 978-3-95450-237-0 ISSN: 2673-5512 doi:10.18429/JACoW-PCaPAC2022-FR022

[6] J. Kreps, N. Narkhede, and J. Rao, "Kafka: a Distributed Messaging System for Log Processing," 2011.

[7] *Daqlite*, 2022. https://github.com/ess-dmsc/daqlite

[8] S. Heybrock, O. Arnold, I. Gudich, D. Nixon, and N. Vaytet, "Scipp: Scientific data handling with labeled multi-dimensional arrays for C++ and python," *J. Neutron Res.*, vol. 22, no. 2-3, pp. 169–181, 2020. doi:10.3233/jnr-190131

[9] *scipp - Multi-dimensional data arrays with labeled dimensions*, 2022. https://scipp.github.io/

[10] M. J. Christensen and T. Richter, "Achieveing reliable UDP transmission at 10 Gb/s using BSD socket for data acquisition systems," 2017. doi:10.48550/ARXIV.1706.00333

[11] O. Arnold *et al.*, "Mantid—Data analysis and visualization package for neutron scattering and μ SR experiments," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 764, pp. 156–166, 2014. doi:10.1016/j.nima.2014.07.029