ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPGW003

HIGH-LEVEL APPLICATIONS FOR THE SIRIUS ACCELERATOR CONTROL SYSTEM

X. R. Resende *, F. H. de Sá, G. do Prado, L. Liu, A. C. Oliveira,
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

Sirius is the new 3 GeV low-emittance Brazilian Synchrotron Light source under installation and commissioning at LNLS. The machine control system is based on EPICS and when the installation is complete it should have a few hundred thousand process variables in use. For flexible integration and intuitive control of such sizable system a considerable number of high-level applications, input/output controllers and graphical user interfaces have been developed, mostly in Python, using a variety of libraries, such as PyEpics, PCASPy and PyDM. Common support service applications (Archiver Appliance, Olog, Apache server, a mongoDB-based configuration server, etc) are used. Matlab Middle Layer is also an available option to control EPICS applications. Currently system integration tests are being performed concomitant with initial phases of accelerator commissioning and installation. A set of functionalities is already available: Linac's control; timing subsystem control; machine snapshots; optics measurements and correction; magnets settings and cycling; Booster orbit acquisition and correction, and so on. From the experience so far, subsystems communications have worked satisfactorily but there has been a few unexpected component misbehaves. In this paper we discuss this experience and describe the libraries and packages used in high-level control system , as well as the difficulties faced to implement and to operate them.

INTRODUCTION

Sirius is the new synchrotron light source of fourth generation and 3 GeV energy, under construction at Brazilian Synchrotron Light Laboratory (LNLS) [1]. It is a MBA-type lattice designed for very low emittance. The machine is in final phases of the Storage Ring installation and beginning of Booster subsystems commissioning. Its control system has been developed over the last few years and it is now being integrated and tested.

Sirius's Linac, build by SINAP [2] has been delivered, commissioned and its control system (CS) successfully integrated in 2018. The Linac-Booster transport line has also been commissioned last year and, at this moment, the first hundred turns of beam in the Booster have been achieved. Alongside these activities, CS architecture validation and components integration were performed, albeit with yet reduced number of installed devices.

In the next sections of this paper we present an overview of control system development and we briefly discuss the EPICS server applications and input/output controllers (IOC) that give support for the high-level applications (HLA). In the sequence we detail the architecture of the HLA and its current development status. Finally we describe how the integration of the CS has been evolving during machine commissioning and end the paper with conclusion remarks on what the next steps are in HLA development and testing.

CONTROL SYSTEM OVERVIEW

The Sirius accelerator control system (SCS) is based on *EPICS* [3], version R3.15. All SCS software components are open-source solutions developed collaboratively using git version control and are publicly available in the Sirius organization page [4] at Github.

The naming system used in Sirius for devices and CS properties is based on ESS naming system [5]. It was adopted and implemented after SINAP had built the Linac, which uses a different system.

Control room desktop configuration and device application deployment are managed with scripts and Ansible automation tool [6].

Although exclusively not the only programming language, *Python 3.6* is employed for most of the software development in the HLAs, and also in a considerable part of soft IOC.

Sirius beamlines control system was developed independently but it has a similar software infrastructure [7].

EPICS SUPPORT SERVICES

Various EPICS support service applications are used in the SCS. All these services run in docker containers to allow for easy testing, deployment and eventual host migration.

Data archiving, for example, is done with *EPICS Archiving Appliance* [8]. Currently there are around 20 thousand process variables (PVs) being archived and ~100 thousand channels, with a storage rate of the order of 36 GB/day. The number of PVs being archived is expected to increase by a factor of ~20 when the CS components of the Storage Ring are added in the near future. The storage rate should go up a more modest factor though, since the most demanding PVs (from Storage Ring BPMs) are already being archived for stress testing purposes.

As for activity logging related to machine installation, operation, component failures and experiments, *Olog* [9] is being used. We are yet to implement a software library that can be used to automate status log insertions and force predefined formatting suitable for automated performance analysis of the machine.

An http *Apache* [10] server is being used to centralize distribution of static information for system components. For example, the flexible CS architecture of magnet power

* ximenes.resende@lnls.br

ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPGW003

supplies is defined in cross-referenced text tables available through this server. Excitation curves of magnets are also available in the same way. This allows for rapid reconfigurations without the need of software update deployments across CS components.

In order to save and restore general machine configurations (machine snapshots, measured and model response matrices, across-desktop HLA window states, and so on) a web service has been written with Python back- and front-ends. The back-end API was developed using the *Flask* [11] framework and it interacts with *MongoDB* [12] database engine using *PyMongo* [13]. The front-end API allows for definition of arbitrary configuration types. Machine snapshot configurations, for example, are defined by template structures and they consist of EPICS PVs name lists.

IOCS

Sirius' control system IOCs were developed using a mixture of EPICS base, module [14] and extension [15] libraries, such as StreamDevice, areaDetector, EtherIP, asynDriver, Motor, procServ, PCASPy [16], and others. Most of the IOCs run as dockerized services.

Linac IOCs were developed by SINAP in Python 2.6 using both PCASPy version 0.6.3 (power supplies) and EPICS base extensions (EGun, Low Level RF, Klystrons, Protection System, etc). Most of the RF subsystem IOCs were based on an initial version from DIAMOND. For other subsystems the IOCs were developed at LNLS. IOCs of vacuum gauges and pumps, temperature sensors, radiation dose acquisition, pulsed magnet electronics, timing system, BPM acquisition control, beam loss monitors and some of RF subsystem components were implemented using EPICS base and its extensions.

Magnet power supply IOCs, on the other hand, were written in Python using PCASPy v0.7.1 and currently run on BeagleBone Black single-board computers (BBB) with 1GHz ARM cores. BBBs communicate with power supply controllers via RS485 using a proprietary protocol [17]. Counting all subsystems' single-board computers, around 150 BBBs have been employed so far, but the number will rise to around 500 when the Storage Ring control is integrated into the system. Additionally, Python soft IOCs were developed to convert power supply currents to normalized magnet strengths. These IOCs run in common computer servers. Finally, there are a few important additional soft IOCs implemented with PCASPy, such as power supply diagnostics, slow orbit feedback correction (SOFB), high-level timing abstraction, optics tune and chromaticity corrections, transport lines beam injection control, and so on.

HIGH-LEVEL GUI APPLICATIONS

As for high-level controls, they are done with GUI applications running in Debian 9 computer desktops. They were written mostly using PyQt5 [18], PyDM [19], that consists in a Python layer on top of Qt [20] and PyEpics [21], and an in-house developed *siriuspy* package. The choice of PyDM, a SLAC [22] initiative with outside contributions, including a few from Sirius staff, allowed for developments concentrated in Sirius-specific issues. It proved to have a smooth learning curve and rapid development using Qt framework's drag-and-drop tools and easy integration with other Python packages.

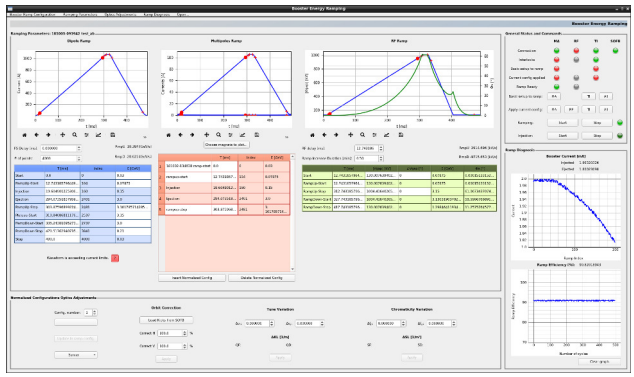

Figure 1: An example of a GUI application: an interface to optimize and control Booster energy ramp process.

The GUI applications set is composed of various windows that can be launched from a main application. Among others, window control components of the GUI collection are: Linac launcher, power supplies, transport lines, BPMs, magnet cycling, Booster ramp, configuration loader, SOFB, etc. Some examples of applications are shown in Figs.1 and 2.

CS-Studio [23] was also used during subsystem developments and some are still currently in use for commissioning, so as to access subsystem's very specific properties. GUI for Linac was developed in EDM [24] by SINAP and it is integrated into the main launcher interface and launched as separate system processes. There are plans in the future to progressively rewrite most of Linac's GUI and CS-Studio applications in PyDM, in order to have a more uniform look-and-feel system.

The choice of using an http server and the *siriuspy* library, thus centralizing many of CS data structures that EPICS clients and servers need, improved code maintainability through reuse and minimizes chances of Python GUI client and EPICS server data inconsistencies.

The Matlab Middle Layer toolkit (MML) [25], integrated with Accelerator Toolbox(AT) version 1.3 [26], is available to control EPICS applications through LabCA [27] and MCA [28]. It is currently being used to simulate and to test model correction matrices in Booster subsystems commissioning. The plan is to use various algorithms already implemented in it, like LOCO and matrices measurements.

SYSTEM INTEGRATION DURING SUBSYSTEMS COMMISSIONING

The first on-site integration tests of in-house HLA developments began in November 2018, with commissioning of Linac-Booster transport line, with relatively few CS components. These initial tests progressed very well, requiring

ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPGW003

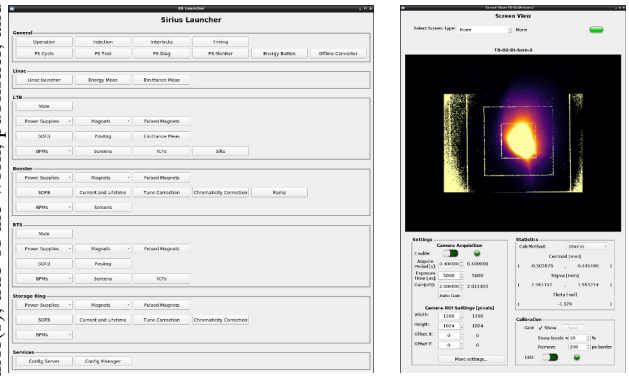

Figure 2: Other examples of applications: the main launcher and a screen device monitor.

only very minor updates of the software, mostly related to the EPICS network environment.

After the shutdown of 2018, CS integration tests resumed in February 2019 with optimization of the Linac and the transport line, and with the initial commissioning of Booster subsystems in March. At this point more components were in use and integration tests enlarged in scope.

In particular, with timing signals now distributed to magnet power supplies (PS), other operation modes of the PS for energy ramp, cycling and for demagnetization were finally tested. These tests revealed unanticipated inconsistencies of IOC behaviour triggered by new BBB-PS architectures that had not yet been tested. These inconsistencies were identified and corrected.

In order to reduce the number of single-board computers employed in the installation park, in some cases many PS are connected to a single BBB that runs the PS IOC. This economy is limiting PV refresh rates that the PCASPy IOC can manage, given the limited CPU resources of single-board computers. IOCs' code upgrades had to take place in order to improve their responsiveness to HLA applications that cycle magnet fields and load machine snapshots. To improve performance, we plan to move the PS IOCs to desktop servers with more computational resources. We can do this without much IOC code refactoring by simply replacing the UART library the IOC currently uses with a TCP/IP library with the same API but connecting to a lightweight server running in the BBB and that takes over the communication with the power supply over RS485. Moreover, IOCs for pulsed magnets, BPMs and timing subsystems also have had minor modifications.

A single network event in mid-April, not yet understood, rendered all PCASPy-based IOCs running in the same subnet, but on different host architectures, unresponsive at the same time. It is speculated that Archiver Appliance broadcasts triggered the event, since it coincided with the moment when a considerable fraction of BPM PVs were disconnected. None of Linac IOCs running in a separate subnet, but were also usgin PCASPy, were affected.

IOCs written in Python and using PCASPy were intentionally designed from scratch so that PCASPy could be easily swapped out for other EPICS CA servers. This now might come in handy if spurious events like the one observed start to happen frequently and we conclude that PCASPy is the culprit.

Another issue we had until recently was with Olog inadvertedly missing entries. It turned out this behaviour was due to an improper choice of storage disk partitioning in the server running the application. Increasing the partition size where application database ran solved the problem.

Generally, the integration tests until now has presented good progress, despite the fact that they were performed concomitantly with the commissioning.

CONCLUSION

Sirius is the new 3 GeV low-emittance synchrotron light source under commissioning at LNLS. Its EPICS based control system is now under integration tests. Overall the integration components has been evolving satisfactorily, given the fact that installation time-line allowed for very limited before-hand integration. All bugs with impeding or severe performance limitation impacts have been promptly identified and fixed. Nevertheless there are a few improvements that are in order for the near future.

We plan to move power supplies IOCs from single-board computers to a desktop server with greater processing capacity, which will solve problems of low update rate and that can be done without major code refactoring. For standardization and better integration of HLAs, we plan to migrate CS-Studio and EDM GUI applications to PyDM, as well as translate Linac PV names to Sirius naming convention.

Due to a spurious event occurred during commissioning that rendered all PCASPy-based IOCs, maybe we will migrate IOCs to another python library, which can rapidly be performed given the structure in which they were developed.

We plan to continue the development of applications to ease commissioning and operation, for example through implementing tools that facilitate debugging and searching correlations between machine parameters. Another future step for CS development is the integration of EPICS support applications to in-house python packages, including scripting implementation of data recovery and analyses from the Archiver, formatted insertion in Olog and generation of machine performance reports.

ACKNOWLEDGEMENTS

We would like to acknowledge the software development group (SOL) of the LNLS beamlines division for pointing out a few years ago to us PyDM as an option for HLA development in Python. In particular we thank L. P. Carmo in the group for the thorough analysis of many HLA architecture options she did in her internship work. We would also like to thank some of the ESS Staff, in particular K. Rathsman, for sharing with us information on ESS naming system and on their web-based naming service.

ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-WEPGW003

REFERENCES

[1] A.R.D. Rodrigues *et al.*, "Sirius Status Update", presented at the 10th Int. Particle Accelerator Conf. (IPAC'19), Melbourne, Australia, May. 2019, paper TUPHW003, this conference.

[2] SINAP - Shangai Institute of Applied Physics, http://english.sinap.cas.cn/

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

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

[5] ESS Naming System, http://eval.esss.lu.se/DocDB/0000/000004/010/ESSNamingconvention_20131011_v2.pdf

[6] Ansible is Simple IT Automation, https://www.ansible.com/

[7] G.S. Fedel, D. B. Beniz, L. P. Carmo, and J. R. Piton, "Python for User Interfaces at Sirius", in *Proc. 16th Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'17)*, Barcelona, Spain, 2017, paper THAPL04, pp. 1091-1097.

[8] EPICS Archiver Appliance, http://slacmshankar.github.io/epicsarchiver_docs/index.html

[9] Olog, http://olog.github.io/2.2.7-SNAPSHOT/

[10] Apache, https://httpd.apache.org/

[11] Flask, http://flask.pocoo.org/

[12] MongoDB, https://www.mongodb.com/

[13] PyMongo, https://pypi.org/project/pymongo/

[14] EPICS Modules, https://epics.anl.gov/modules/index.php

[15] EPICS Extensions, https://epics.anl.gov/extensions/index.php

[16] PCASPy, https://pypi.org/project/pcaspy/

[17] Basic Small Message Protocol, https://github.com/lnls-sirius/control-system-constants/blob/master/documentation/bsmp/protocol_v2-30_en_US.pdf

[18] PyQt5, https://www.riverbankcomputing.com/static/Docs/PyQt5/

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

[20] Qt, https://doc.qt.io/

[21] PyEpics: Epics Channel Access for Python, https://cars9.uchicago.edu/software/python/pyepics3/

[22] SLAC Lab Organization at Github, https://github.com/slaclab

[23] CS-Studio, http://controlsystemstudio.org/

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

[25] G. Portmann, J. Corbett, and A. Terebilo, "An accelerator control middle layer using Matlab" in *Proc. 2005 Particle Accelerator Conference (PAC 2005)*, Knoxville, TN, USA, 2005, pp. 4009-4011. doi:10.1109/PAC.2005.1591699

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

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

[28] A. Terebilo, "Channel access client toolbox for Matlab", in *Proc. 8th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2001)*, San Jose, CA, USA, Nov. 2001, paper THAP030, doi:10.2172/799983