ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

PRELIMINARY DESIGN FOR THE ALBA II CONTROL SYSTEM STACK

Sergi Rubio-Manrique, Fulvio Becheri, Guifre Cuni, Roberto Homs-Puron, Zbigniew Reszela
ALBA-CELLS, Cerdanyola del Vallès

Abstract

One of the main pillars of the ALBA Synchrotron Light Source (Barcelona, Spain) Strategy Plan is the preparation of ALBA to be upgraded to a fourth-generation light source. To accomplish this, a preliminary design of the accelerator has already been initiated in 2021. On the Computing side, the upgrade of the accelerator will require a comprehensive overhaul of most parts of the Control System, DAQ, Timing, and many other systems as well as DevOps strategies. This need for a major redesign will bring new architectural challenges, and opportunities to benefit from new technologies that were not present at the time ALBA was designed and build. This paper presents the preliminary design studies, pilot projects, new approaches to development coordination and management, and the preparation plan to acquire the knowledge and experience needed to excel in the ALBA II Control System Stack design.

INTRODUCTION

ALBA II Preliminary Studies

In 2019 the ALBA Synchrotron Accelerators Division published its first studies [1] on upgrading the existing storage ring to a 4th generation synchrotron lightsource with a low emittance and high brilliance machine; while reusing current injectors, tunnel and building. ALBA II White Paper [2] was finally released in May 2023, being the construction and commisioning of the new machine expected between 2028 and 2031.

ALBA's Computing and Controls Division have developed its own ALBA II preliminary study [3], structured in 11 different areas: Input/Output controller architecture, power supplies, timing system, equipment protection system, personnel safety system, IT architecture, motion control, control system stack, configuration management and stock management, machine learning, realtime processing needs.

This study focuses on the common software of the accelerators and beamlines control. Experiments control needs are explored in a separate article [4].

Control System Challenges of ALBA II

ALBA II poses unique challenges as a 4th Generation Synchrotron Light Source. It demands smaller, more stable photon beams and a greater number of independently controlled magnets, using faster orbit feedback loops. Faster RF control loops and precise diagnostic elements are essential, especially for nano-focus beamlines.

To ensure control system alignment with final specifications, we started preliminary studies to foresee and evaluate the technologies required for ALBA II.

Current Status of ALBA Control System

The ALBA Control System (CS) Stack consists of thousands of Tango [5, 6] Device Servers controlling hardware and software processes, user interfaces like Taurus GUI [7, 8] and distributed Tango Control System services (Archiving [9], Alarms [10], Sardana [11], etc.)

Recognizing component obsolescence, an upgrade began in 2019 [12], migrating to modern technology (64-bit CPUs, Tango 9.3, Debian Linux 10, Python 3.x) and applying methodologies that guarantee a continuous evolution of the stack: continuous integration and delivery pipelines, guided incremental change, continuous delivery and testing, fitness functions measurement and focus on performance. Paving the way to the ALBA II transition.

Control System Upgrade Work Packages

The ALBA II control system upgrade requirements are being categorized into four distinct areas:

- Distributed System & Event. Related to IOCs, timing, IT infrastructure and real time loops.
- Tango Control System. New features to be introduced in our control framework.
- Graphical User Interfaces. Addressing new security and performance requirements, along with web technologies.
- DevOps: software development, packaging, integration, deployment and operation. Related with IOCs, IT Architecture and Configuration Management.

DISTRIBUTED CONTROL SYSTEM

Preliminary Study

A study of all communication processes have been started in order to detect control system issues (event overload, communication bottlenecks, RAM/CPU issues), and to improve the current interaction between accelerators and beamlines via gateway machines.

Our investigation identified bottlenecks mostly in the magnet power supplies subsystem, the one that will become the most critical in the upgrade to ALBA II. The problems that have been identified are:

- Dependency in old hardware and legacy event system (notifd) that limits the event rate.
- Bursts of events saturating applications.
- Slowness in event subscription due to the high number of attributes accessed by applications.
- CPU peaks causing python memory leaks.

Several strategies have been implemented to minimize these problems: upgrade of most control hosts, monitoring and restart of legacy event daemons, reconnection mecha-

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

nisms on UIs, composer and gateway devices to redistribute connections, delayed event subscribing, ... Strategies to be tested against the needs for ALBA II.

Those needs include higher number of devices, larger data sizes, refresh frequencies at 10 Hz, nanoseconds timestamps. To match this control performance and enable machine learning it is required a redesign of the event propagation in the non-real time control system to reduce latencies down to 100 ms.

Technical Proposals of Improvement

Building upon experiences from other 4th generation Synchrotron Lightsources using Tango and Python as their primary control system framework, we have outlined several improvements to address our current challenges. Some of these enhancements will be incorporated into the Tango kernel, while others will be part of our development efforts. These include:

- Use of just-in-time compiler in python (numba)
- Upgrade to CPU-optimized versions of python
- Usage of C++ background threads
- Event filtering prior to processing.
- Channeled access via composer/gateway devices.
- Queuing, caching and buffering on middle-layer devices to manage bursts.
- Redistribute devices among processes and hosts.
- Use profiler tools to control CPU/RAM usage.
- Unsubscribe attribute events on hardware error.
- Minimize code execution in repetitive hooks.
- Use direct-event-channels between processes.

These solutions to have better control over event throughput will be integrated into core libraries and development guidelines. Core libraries and abstract classes will also help in reducing vendor-dependance and hardware obsolescence on DAQ systems.

Additionally, we will implement security measures at the Tango kernel level, with careful assessment of their impact using fitness functions to prevent regressions.

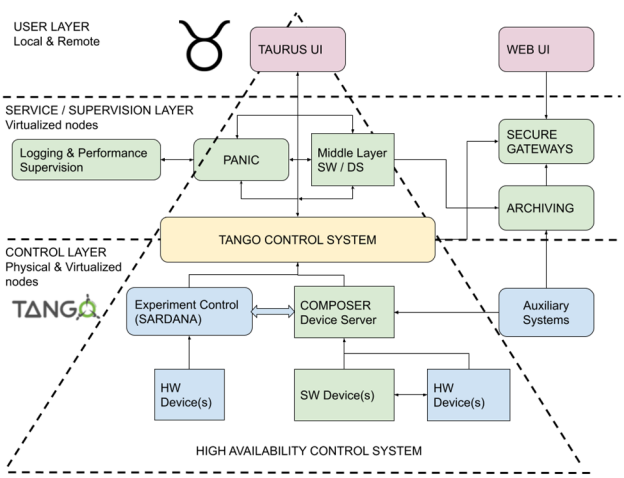

Figure 1: ALBA II Control System architecture.

Split into Critical vs Non-Critical Architecture

One of the keys for our success in upgrading the ALBA Control System has been the clear separation between subsystems and between critical hardware-based control systems and virtualized supervision systems (Fig. 1).

This clear separations allowed us to gradually upgrade systems and layers separately. This successful approach will be kept in the new control system, in which a much virtualized system may imply a higher risk of excessive interdependence between operation and supervision.

TANGO CONTROL SYSTEM

Tango Control System 9.3 [13] runs several 4th Generation Synchrotron Lightsources, either in operation (ESRF-EBS, MaxIV) or being upgraded (Soleil, Elettra).

Some of the features that enabled Tango success are:

- A large and dynamic community, providing shared experience and a common effort for development.
- Archiving and Alarm services, providing laboratories with standardized and reliable tools.
- Python and matlab bindings, enabling accelerator scientists and control developers to share code and simulations during the design phase.
- The Tango Device Server Catalog, providing hundreds of already developed device classes.

In addition to these core features, recent improvements enabled faster archiving systems, python-based event processors and façades, database gateways, device access control, container images and microservices, high level python API's, web applications and better tools for configuration and tuning of the control system.

Room for improvement exists regarding performance and security of the control system. Since 2021 the Tango Collaboration has triggered the RFC [14] project, to standardize and document the control system features, and Special Interest Group Meetings (SIG); where panels of experts meet to study and design future Tango features.

Tango 10 and beyond

SIG meetings [15] have discussed performance, CI/CD, documentation, alarms integration, event filtering, graphical user interfaces, archiving ... These meetings have collectively shaped a roadmap for upcoming Tango releases, gradually introducing new features [16].

ALBA plays an active role working on these topics: faster subscription and reconnection, serialization model, background threads, reliable polling, asynchronous communication, event buffering and filtering. In addition to performance issues, we identified the need to increase the security of the system, assuming that web-technologies are going to take a much bigger role in the control of ALBA II.

We foresee the need of more reliable and secure facade device servers acting as gateways/firewalls, improving the Tango Access Control service as a complementary layer to current IT security technologies already in place.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

A New Archiving for ALBA II

ALBA's current HDB++ Archiving System, in operation since 2019, stores 21,000 attributes for the accelerators using MariaDB. It employs a single database host and 4TB SSD disk for storing the 1 year raw data, along with another host for long-term decimated data.

While suitable for current needs, ALBA II presents specific requirements, like event frequency and timestamp resolution, beyond the our current seup capabilities. As well as the need of a high-availability cluster.

The proposed solution involves deploying a TimeScaleDB cluster, based on the actual system in production at the ESRF-EBS Control System [17]. However, this upgrade may incur significantly higher costs. To ensure a smooth and cost-effective transition to TimeScaleDB, a pilot project is planned for the next year; aimed at building expertise in configuring and optimizing TimeScaleDB databases.

GRAPHICAL USER INTERFACES

The vast majority of GUIs at ALBA are based on Taurus [18], a library for developing graphical and command line client applications. Designed for controlling and monitoring the Tango control systems, it has undergone thorough refactoring, and now it is built upon a robust, data-source-agnostic core that supplies data to generic widgets in which new data sources can be integrated into Taurus by developing scheme plugins.

Developing the Taurus library in Python and basing it on PyQt and PyTango bindings has proven to be a great design decision. Ensuring an easy and stable development framework and flexibility to allow for nearly any level of interface customization thanks to the TaurusGui framework [19] and its extensive catalog of widgets. The success story of Taurus has attracted other institutes to adopt it, and it has become the default solution for developing Tango GUIs in Python, now a community-driven project.

For ALBA II, Taurus remains the choice for desktop applications. Efforts to enhance scalability, performance, and simplify configuration architecture are ongoing. A performance optimization project has been started in collaboration with the ESRF and the Taurus community [20], as well as multiple efforts to improve the configuration of applications and the portability of applications between different OS versions and platforms.

Web-Based Applications

Web-based and mobile applications are rapidly gaining popularity in control rooms of installations similar to ALBA. This technology should be carefully evaluated as a possible complement or even alternative to Taurus desktop apps. Running operators' GUIs on mobile devices offers the obvious benefit of portability and ease of use, allowing operators to access instruments for visual inspection while simultaneously using a familiar control and monitoring interface.

After conducting a brief survey of available web solutions, we have pre-selected Taranta and JupyterLab . Using Taranta[21] out of the box, one can easily navigate through a Tango database and select a device for monitoring. Taranta widgets allows to create custom dashboards, resembling Taurus flexibility. On the backend side, Taranta uses TangoGQL with a powerful query language that aligns well with the organisation of the Tango Database.

Another general-purpose web application we identified is JupyterLab [22], which could serve specific needs, such as beamline experiments or controlling and monitoring accelerator operation scripts. Tango attributes can be easily monitored from Jupyter notebooks using the jupyTango project [23].

ALBA II user interfaces will continue using Taurus desktop applications mostly for performance and security reasons. Nevertheless, we will need to gain experience in the highly specialised skills needed for frontend development of web applications and solutions and to them when desktop solutions cannot fully meet user requirements.

SOFTWARE PACKAGING INTEGRATION AND DEPLOYMENT (DEVOPS)

Considering the ALBA CS Stack maintenance experience and looking at the far future of its successor we believe that the later one should strive for being highly modular and kept up-to-date. This will be empowered by practices such as Continuous Integration, reliable software configuration, deployment and monitoring, among others; and technologies like Docker containers, Kubernetes orchestrator, conda environments, etc.

Current Status

Several strategies had proven success at ALBA:

- Replacing custom solutions with generic mechanisms, such as Python packages instead of scripts, and creating distribution packages (Debian, conda) with Gitlab CI for efficient deployment.
- Implementing reproducible and automated processes using Gitlab CI for testing, quality checks, distribution packages, and documentation.
- Facilitating testing and rollbacks through update schedules, pre-production machines, conda environments and staging repositories for canary releases.
- Centralized administration of accelerator diskless PCs using a single linux image for all hosts.

But, we also identified several weaknesses:

- Lack of well-defined policies and quality rules hinder collaboration and lead to local optimizations instead of general solutions.
- The absence of an authority to enforce policies and quality requirements can result in deviations.
- Lack of continuity in keeping systems up-to-date, including lengthy update cycles, non-testing strategies, and limited end-user involvement.
- We do not apply enough best practices in configuration management, like implementing APM and KPI for control applications and services or using in-premises CI infrastructure.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

Technical Proposals in DevOps

Considering the potentially vast scope, we focused on the following main topics identified as the main ones in need of improvement:

- Coordination strategies to maximize efficiency, including knowledge sharing, team experts with newcomers, standardized toolsets, external priority setting, and incremental development.
- Keeping production systems up-to-date through configuration management tools, automation, easy rollbacks, packaging simplification, and deployment isolation.
- Gathering feedback on control system performance through automated logging, monitoring (centralized logging, Prometheus, alarms), and human perception (closely working with end-users, product owner validation, and metrics compilation).

Addressing Hardware Obsolescence at ALBA

Data Acquisition (DAQ) systems are integral to ALBA's Control System, serving critical roles in various applications. However, these hardware-based DAQ and synchronization systems present challenges during system upgrades. Adapting drivers to new operating systems, migrating from 32-bit to 64-bit platforms, and rewriting C++ device servers for updated libraries are common obstacles.

To mitigate these challenges, a more flexible approach could have been adopted earlier by interfacing hardware elements using abstract classes. This approach would have allowed seamless replacement of old hardware with new models or vendors, requiring only adjustments to the device server's lower side. Clients and control system configurations would remain unaffected.

ALBA initially explored this approach, aiming to reuse classes across institutions and combat hardware obsolescence. Benefits of interfacing with abstract classes include:

- Vendor-Independence: Avoiding vendor lock-in.
- API Reuse: Leveraging existing APIs.
- Hardware and GUI Independence: Upgrading hardware and GUI components independently.

Similar constraints affecting other vendor-locked systems like EPS and PSS, which rely on PLCs, can be addressed through the adoption of industry standards like OPC-UA. This standardization allows integration of various PLC brands and facilitates hardware replacement without disrupting the existing architecture.

PILOT PROJECTS

As a result of the presented studies, several Pilot Projects plan have been scheduled encompassing a range of objectives and tasks aimed at enhancing the ALBA Control System functionality and infrastructure:

- Always-Ready CS Stack: Ensuring a stable, easily updatable CS stack by automating maintenance, continuous integration, and testing workflows.
- Conda Evaluation: Exploring Conda and virtual-environment solutions to decouple the CS stack from the underlying OS.
- Centralized Logging: Implementing the Elastic Stack for centralized logging of Sardana service.
- Automatic Testing: Introducing automatic testing using real hardware in CI pipelines.
- Kubernetes Integration: Extending Kubernetes usage for hardware interaction.
- Configuration Management System with rollback mechanisms and better application configuration.
- Technology Benchmarking: Evaluating technologies using performance metrics.
- Benchmarking Current CS: Collecting performance data from the existing ALBA CS and assessing system reliability.
- Obsolete Hardware/Software Replacement: Exploring alternatives for outdated components.
- Taranta Deployment: Deploying Taranta in a Kubernetes cluster for application deployment.
- Web Technologies Proficiency: Developing proficiency in web technologies.
- Jupyter Lab Skills: Mastering custom Jupyter Lab extensions and Plotly integration.
- Taurus Performance Improvement: Addressing Taurus application performance issues.
- PyTango Device Servers Monitoring: Monitoring cpu, memory, threading and logging.
- Database Evaluation: Redis, InfluxDB, TimeScaleDB as the new archiving engine.

NEW APPROACHES TO DEVELOPMENT COORDINATION AND MANAGEMENT

The ALBA Control Section, 16 software engineers and 4 automation engineers, is organized into four groups: accelerator controls, beamlines controls, software stack, and automation. They support 23 customer units and participate with other institutes in projects like Icepap, Tango, Sardana, and MXCuBE.

The ALBA Controls Section was previously organised as follows: engineers were designated as 'controls contacts' for one or more customer units and inter-institute collaborations to fulfil service requirements. Simultaneously, engineers worked in small teams following the Scrum framework for new developments. The time allocation for these roles was approximately 60 %-40 % [24]. This approach served us well for several years. However, considering that the ALBA II project will run in parallel with ALBA's ongoing operations, we anticipate a need for even better organisation and efficiency. Capitalising on the significant turnover we experienced in the past year when over 50 % of the section left ALBA, we decided to reorganise our work processes [25] as follows:

- Introduction of 'Service Support': A new role, 'service support,' rotates daily, focusing on handling new requests from customer units. Requests are classified, refined, and potentially moved to the development backlog if they require new development work. Service support tasks are prioritized and visualized on a Kanban board.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

- Development Board: Engineers not in the service support role engage in development work using individual backlogs (one per customer unit and component).

The ALBA section/group benefiting from a development prioritises the tasks. Tasks are pulled onto the Kanban development board during planning events after serialisation of multiple backlogs into one team backlog. There are only a few exceptions where work can enter the board without planning, such as high-priority incidents that cannot be resolved by the service support.

This new approach enables us to work at a sustainable pace while maintaining maximum transparency, separating service support duties and allowing the rest to focus on development work. The controls contact role have been maintained but its duties limited to refinement of the backlog.

We achieved just-in-time planning with a single source of upstream work, while leaving room for unplanned and expedited tasks to be added to the board. Limiting work in progress allows us to concentrate on work quality, thanks to explicit design and review phases that complex tasks must undergo. However, we anticipate that it will take time to refine and transition from the old processes.

CONCLUSION

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 and the whole technology stack 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. The big turnover in the team composition when transitioning from the development into the operation phase was probably decisive in postponing this effort.

This paper only explores some aspects of the control system. Other work packages that are completely interrelated are the EPS and PSS protection systems, the Experiment Control Framework, Motion Control and Machine Learning, as well as the election of our future IOC platform. Although those developments will run along the main control-system stack, it's development phase will follow a parallel path due to the differences in specifications, platform and/or implementation phase.

Some of the key decisions to be taken when designing ALBA II will be conditioned by our current struggling with hardware obsolescence in all those projects. Abstract classes, virtualization and other strategies will be required to avoid vendor-locking situations that will block future improvements of the machine. A vision towards and evolutionary architecture and the need of continual improvement will be needed in order to guarantee the long-term operation of the control system that we are just starting to design.

The result of this preliminary study is not a static document, but a commitment to continue following and exploring 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 Section members. These projects would not be possible without support from other sections of the Computing Division, mainly IT Systems Section, we would like to acknowledge Sergi Puso for his support in multiple fields.

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] F. Pérez *et al.*, "Alba II Accelerator Upgrade Project", in Proc. IPAC2022, Bangkok, Thailand, pp. 1467-1470. doi:10.18429/JACoW-IPAC2022-TUPOMS027

[2] ALBA II White Paper, https://www.cells.es/en/science-at-alba/alba-ii-upgrade/alba-ii-whitepaper.pdf

[3] O. Matilla *et al.*, "Towards the ALBA II : the Computing Division Preliminary Study", presented at ICALEPCS'23, , Cape Town, South Africa, Oct. 2023, paper TUPDP077, this conference.

[4] Z. Reszela *et al.*, "Improving User Experience and Performance in Sardana and Taurus: A Status Report and Roadmap", I presented at ICALEPCS'23, , Cape Town, South Africa, Oct. 2023, paper THPDP050, this conference.

[5] TANGO - An Object Oriented Control System Based on CORBA, https://www.elettra.eu/icalepcs99/proceedings/papers/wa2i01.pdf

[6] Tango, https://tango-controls.org

[7] 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-THHC3O03

[8] Taurus, https://taurus-scada.org/

[9] D. Lacoste *et. al.*, "New developements on HDB++, the high-performance data archiving for Tango Controls", in *Proc ICALEPCS'23*, this proceedings.

[10] S. Rubio-Manrique *et al.*, "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

[11] 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.

[12] G. Cuni *et al.*, "ALBA Controls System Software Stack Upgrade", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 222-229. doi:10.18429/JACoW-ICALEPCS2021-MOPV037

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP076

[13] A. Götz *et al.*, "The Tango Controls Collaboration Status in 2021", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 544–549. doi:10.18429/JACoW-ICALEPCS2021-WEAR01

[14] V. Hardion *et al.*, "Tango Control RFCs", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 317-321. doi:10.18429/JACoW-ICALEPCS2021-TUBL03

[15] https://indico.tango-controls.org

[16] T. Juerges *et al.*, "The Tango Controls Collaboration Status in 2023", presented at ICALEPCS'23, Cape Town, South Africa, paper TH1BCO03, this conference.

[17] R. Bourtembourg *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. 1117. doi:10.18429/JACoW-ICALEPCS2019-WEPHA020

[18] https://taurus-scada.org/docs.html

[19] S. Rubio *et al.*, "Unifying all Tango Services in a single Control Application", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1052-1055. doi:10.18429/JACoW-ICALEPCS2015-WEPGF148

[20] https://gitlab.com/taurus-org/taurus-followup

[21] TARANTA: Tango on Web, https://taranta.readthedocs.io/en/latest

[22] https://jupyter.org/

[23] https://gitlab.com/tango-controls/jupyTango

[24] G. Cuni *et al.*, "Introducing the SCRUM Framework as part of the Product Development Strategy for the ALBA Control System", in Proc. ICALEPCS'15, Oct. 2015, Melbourne, Australia, Oct. 2015, pp. 60-63. doi:10.18429/JACoW-ICALEPCS2015-MOD3O04

[25] G. Cuni *et al.*, "A Successful Emergency Response Plan: Lessons in the Controls Section of the ALBA Synchrotron"ALBA Synchrotron", presented at ICALEPCS'23, Cape Town, South Africa, paper TU2AO03, this conference.