6th North American Particle Accel. Conf. (NAPAC2025), Sacramento, CA, USA JACoW Publishing
ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP003

AI-READY CONTROL INFRASTRUCTURE FOR CYCLOTRON SYSTEMS USING GPU-ACCELERATED PYTHON GUIS AND LABVIEW OVER ZEROMQ

C. Lopez*, G. Soto, M. Backfish, I. Lopez, R. Sahebzada, M. Novotny, H. Berns, E. Prebys†
Crocker Nuclear Laboratory, University of California, Davis, CA, USA

Abstract

We present a modular, high-performance control and monitoring framework for the 76-inch isochronous cyclotron at the Crocker Nuclear Laboratory. The system integrates GPU-accelerated Python graphical interfaces with deterministic LabVIEW control running on CompactRIO hardware, connected through asynchronous ZeroMQ messaging. Real-time data acquisition, actuator control, and feedback are complemented by an independent SQLite-based logging process, achieving sustained 100 Hz multi-channel writes without impacting GUI responsiveness. The architecture leverages Python multiprocessing to decouple rendering, communication, and logging tasks, ensuring sub-millisecond end-to-end latency under load. While AI-based optimization is not yet deployed, the hardware, software, and data infrastructure are explicitly designed to enable future machine learning inference and autonomous control without structural changes. This work establishes an AI-ready foundation for digital transformation and data-driven optimization of cyclotron operations.

INTRODUCTION

Modern particle accelerators demand increasingly sophisticated control systems capable of high-speed feedback, reliable data acquisition, and scalable interfaces for long-term adaptability. The Crocker Nuclear Laboratory at the University of California, Davis, operates a 76-inch isochronous cyclotron that has historically relied on fully analog control systems. These systems, while robust, present limitations in flexibility, observability, and compatibility with modern optimization and AI-assisted control methodologies [1,2].

To address these challenges, we have developed and deployed a novel, hybrid control infrastructure that integrates real-time deterministic control via LabVIEW [3] and CompactRIO hardware with GPU-accelerated Python graphical interfaces [4]. The new architecture allows high-resolution control and monitoring of beamline components, while maintaining compatibility with the existing analog hardware via galvanic isolation and custom DAQ integration. Real-time communication between system layers is enabled through ZeroMQ, allowing decoupled asynchronous messaging between LabVIEW and Python processes.

The architecture emphasizes modularity, responsiveness, and extensibility, enabling concurrent real-time GUI interaction, deterministic hardware control, and data logging. While this work does not present machine learning inference directly, it establishes the foundational infrastructure necessary to support such AI-driven decision-making systems [5].

* cslopez@ucdavis.edu
† eprebys@ucdavis.edu

SYSTEM ARCHITECTURE

The system architecture follows a modular four-layer design, encompassing the Field, Control, Supervisory, and Application layers. Each layer is physically and functionally decoupled, allowing for reliable operation, simplified troubleshooting, and future scalability. Figure 1 illustrates the complete signal flow and data processing across the architecture.

Field Layer

This layer serves as the direct interface to the cyclotron's analog infrastructure, including power supplies, electromagnets, sensors, and beam diagnostic instruments. Both analog and digital I/O channels are connected through National Instruments (NI) C Series modules (NI-9264, NI-9228, NI-9205, NI-9403) [6], with all signals routed through custom-built galvanic isolation [7] boards. This approach enables non-intrusive signal acquisition from legacy analog meters, preserving their role as reference indicators in the existing control system while allowing the new digital system to access the same information in real time.

A custom ammeter supports current measurements from the picoamp to microamp range, producing both an analog voltage output and digital range encoding. This feedback is passed to the Control Layer to support precise, real-time beam current monitoring.

Control Layer

This layer runs real-time deterministic logic on a CompactRIO platform with LabVIEW RTOS, managing signal generation, ramping, on/off toggling, and continuous acquisition at fixed intervals (1–10 ms). Interfacing directly with the Field Layer via NI C Series modules, it ensures precise timing and stable beamline control. All state variables are passed to the Supervisory Layer for integration with higher-level operations and potential AI-driven optimization.

Supervisory Layer

Operating on a Windows-based workstation running LabVIEW, the Supervisory Layer functions as the bridge between deterministic hardware control and the Application Layer's GPU-accelerated user interfaces. It uses a Ze-

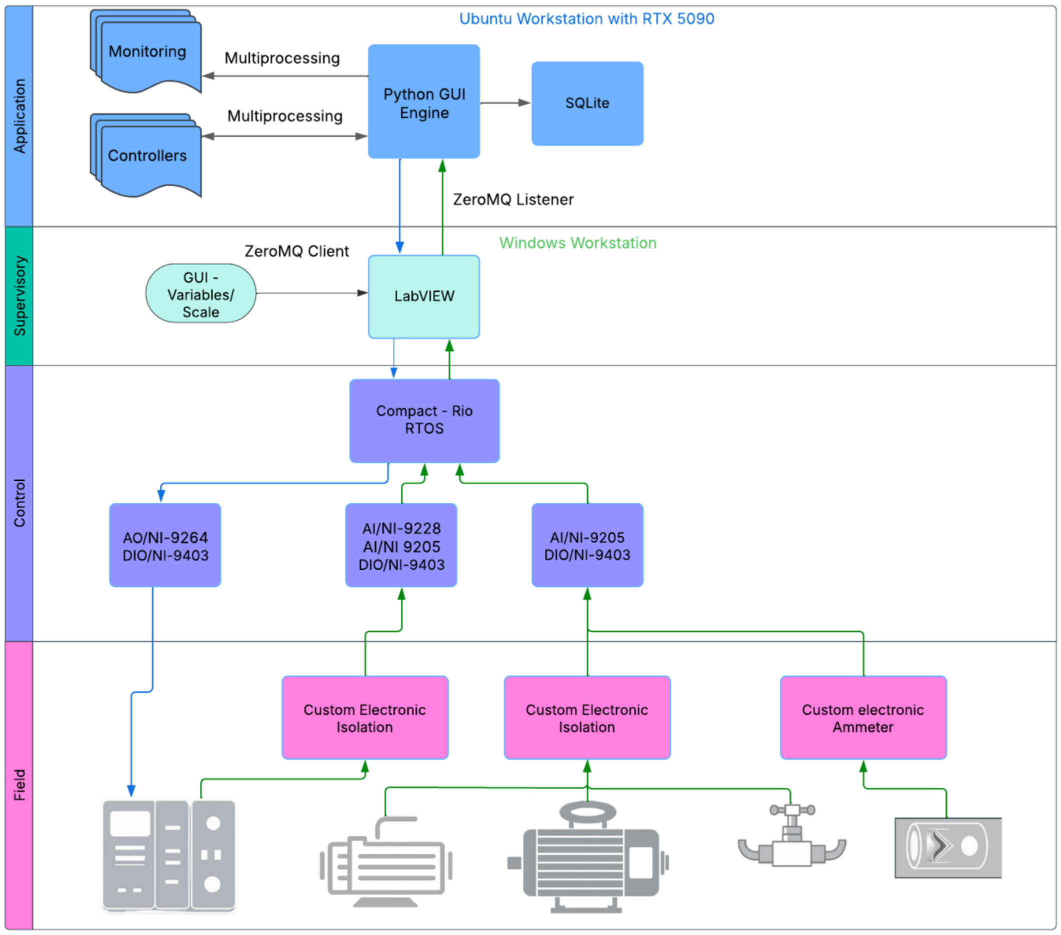

Figure 1: Layered control architecture integrating CompactRIO, LabVIEW, and GPU-accelerated Python GUIs via ZeroMQ.

roMQ [4] client to publish and subscribe to live control data, ensuring low-latency, bidirectional communication with the Python-based GUIs.

This layer also manages operator commands, and scaling of control variables. By centralizing these supervisory functions, it isolates hardware-facing code from GUI and data logging processes, reducing the risk of system instability and simplifying troubleshooting. Its modular design allows new monitoring points, control variables, or AI-driven control loops to be incorporated without modifying the Field or Control Layers.

Application Layer

The Application Layer runs on a high-performance Ubuntu workstation equipped with an NVIDIA RTX 5090 GPU and supports concurrent graphical processes through Python's multiprocessing library, as outlined in Table 1. The Python GUI Engine hosts both Monitoring and Controller windows, each operating in independent processes. A ZeroMQ REP server receives LabVIEW updates and transmits setpoints or control commands in response.

This layer logs data asynchronously via a dedicated SQLite and includes a WebSocket-compatible mobile interface for remote device access. It is explicitly designed to accommodate future AI optimization and autonomous control modules by leveraging the available GPU resources [8].

REAL-TIME GUI AND MULTIPROCESSING DESIGN

The Python GUI engine runs on an RTX 5090-equipped Ubuntu workstation using PyQt6 and Matplotlib for visualization. The application structure separates visualization, messaging, and control logic into distinct processes and threads using Python's `multiprocessing` library.

A dedicated ZeroMQ REP server receives 128-byte binary packets from LabVIEW, each containing a timestamp, 14 channel readings, and a 28-bit control bitmask. GUI processes consume data asynchronously through a thread-safe queue and update double-bar plots (current vs. target) with 100 ms refresh. Additional components display binary status flags and latency diagnostics in real time.

The system's multiprocessing design ensures that rendering, communication, and user interaction remain non-blocking. GUI elements scale with channel count, and future

Table 1: Application Layer Workstation Specifications

Component	Specification
Processor	AMD Ryzen 9 9950X3D (16-core, 32-thread)
GPU	NVIDIA RTX 5090 (32 GB GDDR7, MSI Vanguard SOC)
RAM	64 GB DDR5
Cooling	Corsair iCUE Link Titan 360 mm AIO
Operating System	Ubuntu 22.04 LTS
Purpose	Python GUI engine, multiprocessing visualization, ZeroMQ listener, mobile interface

expansion toward AI-assisted closed-loop control is fully supported by the architecture.

Asynchronous Data Logging via SQLite

To complement real-time monitoring, a dedicated SQLite logging process is implemented using Python's `multiprocessing` and `sqlite3` libraries. Channel data, control bitmasks, and epoch-synchronized timestamps are asynchronously streamed from the main ZMQ listener via a shared queue to the logger process. Benchmarks confirm the system can reliably record over 100 samples per second across 14 channels without bottlenecks or dropped records. This persistent, structured logging enables downstream tasks such as anomaly detection, ML training set generation, and remote diagnostics. The logger operates independently of GUI rendering and control loops, ensuring data integrity even during high-frequency operations.

LATENCY CHARACTERIZATION

End-to-end latency was measured by embedding an epoch-corrected timestamp in each data packet sent from LabVIEW and comparing it to Python's real-time reception. The system achieves a consistent 0.7–0.9 ms latency, including packet decoding, queue handling, and GUI update scheduling.

This sub-millisecond latency confirms the infrastructure's suitability for high-frequency beam control, allowing real-time feedback loops and AI-in-the-loop extensions. No latency spikes or GUI stalling were observed during testing at 1 kHz communication intervals.

DISCUSSION

The presented infrastructure demonstrates how a hybrid control stack can integrate analog hardware, real-time DAQ, GPU-accelerated GUIs, and message-based communication without compromising latency. Its modular design ensures maintainability and allows integration of AI optimizers.

CONCLUSION AND OUTLOOK

This work demonstrates a fully integrated control stack uniting **real-time actuators**, **comprehensive monitoring**, **low-latency feedback**, and **scalable logging** in a GPU-accelerated, multiprocessing-enabled architecture. The system operates reliably at sub-millisecond latency and logs high-resolution, time-synchronized datasets in parallel with live control.

With all components—deterministic hardware control, rich sensing, scalable storage, and compute acceleration—already in place, the system is **AI-ready**. Future work will extend this infrastructure to deploy machine learning algorithms for optimization and autonomous control, starting with offline training and progressing to live beam tuning in production operations.

ACKNOWLEDGMENTS

This work was supported by the Crocker Nuclear Laboratory and the University of California, Davis. Additional support was provided by the ANID Becas Chile scholarship.

REFERENCES

[1] Crocker Nuclear Laboratory, `https://crocker.ucdavis.edu`

[2] C. Lóóez Osses *et al.*, "Automated control and monitoring system for the Crocker Nuclear Laboratory cyclotron", presented at IPAC'25, Taipei, Taiwan, Jun. 2025, paper THPS113, to be published.

[3] E. Loftin *et al.*, "Magnet mapping at LANSCE", in *Proc. IEEE Pulsed Power Conf. (PPC)*, San Antonio, TX, USA, Jun. 2023, pp. 1–4. `doi:10.1109/PPC47928.2023.10310771`

[4] H. Chen, G. F. Liu, D. Zhang, T. Qin, and X. K. Sun, "A study of performance comparison of databases for HALF data archiving system", *J. Instrum.*, vol. 18, no. 12, p. 12015, Dec. 2023. `doi:10.1088/1748-0221/18/12/P12015`

[5] A. Scheinker, "cDVAE: VAE-guided diffusion for particle accelerator beam 6D phase space projection diagnostics", *Sci. Rep.*, vol. 14, p. 29303, 2024. `doi:10.1038/s41598-024-80751-1`

[6] *LT1012 precision input current, microvolt offset, low noise op amp*, Linear Technology, Milpitas, CA, USA, 2018. `https://www.analog.com/media/en/technical-documentation/data-sheets/lt1012.pdf`

[7] *AMC3301 precision, isolated amplifier datasheet*, Texas Instruments, Dallas, TX, USA, May 2025. `https://www.ti.com/lit/ds/symlink/amc3301.pdf`

[8] R. J. Shalloo *et al.*, "Automation and control of laser wakefield accelerators using Bayesian optimization", *Nat. Commun.*, vol. 11, no. 1, p. 6355, Dec. 2020. `doi:10.1038/s41467-020-20245-6`