Building an Open-Source PID Control System for Cryogenic System

1st Nutthapat Pongtanyavichai
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
6532068721@student.chula.ac.th

2nd Poopha Suwananek
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
6532141821@student.chula.ac.th

3rd Monthawat Sawarak
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
6532143021@student.chula.ac.th

4th Kamonluk Suksen*
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
kamonluk@cp.eng.chula.ac.th

5th Prabhas Chongstitvatana
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
Prabhas.C@chula.ac.th

***Abstract*—This paper presents the development of an open-source, laboratory-scale platform for cryogenic system control that utilizes proportional–integral–derivative (PID) controller with remote monitoring through a web interface. Our proxy system consists of a Raspberry Pi 5 microcontroller, a Lake Shore Model 240 temperature monitor, and, polyimide resistive heaters, within a modular API-driven software architecture that separates hardware adapters from system services. We validate the platform via an insulated chamber prototype and a copper thermal conductor prototype. These system proxies emulate qualitative characteristics of cryogenic thermal dynamics (such as long time constants and measurement delays) rather than operating at true cryogenic temperatures. They demonstrated stable closed-loop tracking to a temperature setpoint with bounded overshoot and small steady-state error. The proposed design provides a generalization of various cryogenic setups and a remote control interface which can be extended toward more advanced endeavors relevant to superconducting-qubit laboratory infrastructure.**

***Index Terms*—PID control, Temperature control, Cryogenic systems, Open-source software, Laboratory automation, Remote monitoring.**

I. Introduction

Cryogenic temperature control is a critical engineering problem in modern physics laboratories. Many experiments require a stable low-temperature environment over a long period, with tight tolerances and continuous monitoring. In practice, this means integrating temperature sensors, heaters, and controllers into a closed-loop system that can reject disturbances, handle slow thermal dynamics, and remain operable and accessible on the fly.

In professional research, these capabilities are provided by specialized cryogenic infrastructure and diverse instruments (e.g., temperature controllers, thermometers, flow meters, etc.). Such commercial systems are expensive, proprietary, and not always accessible for early-stage prototyping. While some open-source solutions exist, they often lack a unified, modular software architecture that supports web-based monitoring and standardized hardware interfaces. This limitation restricts hands-on opportunities for students and researchers to learn cryogenic instrumentation, control, and integration.

This paper presents an open-source, laboratory-scale proxy platform for web-based cryogenic monitoring and PID temperature control. It is important to note that our experimental platform is not a true cryogenic system; rather, it is a thermal proxy testbed designed to emulate some qualitative characteristics of cryogenic dynamics such as long time constants, measurement delays, and varying heat loads at accessible, above-ambient temperatures. The main challenge is achieving stable real-time control while unifying devices with non-standard protocols into a consistent software interface. While motivated by superconducting-qubit infrastructure, this robust, user-friendly, and remotely accessible design is broadly applicable to other thermal testbeds.

The main contributions of this work are: a **modular API-driven architecture** that decouples hardware adapters from control logic for flexible instrument integration; and an **end-to-end proof of concept** via a low-cost thermal testbed that demonstrates stable closed-loop PID control, data acquisition, and web-based remote monitoring.

II. Background

Superconducting-qubit quantum computing requires a stable ultra-low-temperature environment; qubits must operate at millikelvin temperatures [1] inside a *dilution refrigerator* (DR) that uses ^{3}He/^{4}He mixtures to continuously extract heat through multiple temperature stages [2], [3]. Maintaining this environment demands precise, automated closed-loop temperature control over extended durations, a task currently served by expensive proprietary systems that are difficult to study, modify, or adapt. This gap motivates the need for an open-source, accessible control platform that allows researchers and students to understand, experiment with, and extend cryogenic control software. In this work, we therefore

*Corresponding author

built a laboratory-scale proxy testbed that reproduces the key operational challenges of such environments, heater actuation, slow thermal dynamics, long-duration stable control, and remote supervision—at a cost and scale accessible for education and early-stage prototyping, with a software stack directly transferable to superconducting-qubit research infrastructure.

We adopt an open-source approach to improve extensibility and reproducibility: all source code is publicly available, enabling students and researchers to replicate, modify, and extend the platform. The system is regulated by a *proportional–integral–derivative* (PID) controller—a classical feedback control algorithm that continuously adjusts its output based on the difference between a desired setpoint and the measured process variable. The PID controller computes the heater command from the error $e(t) = T_{\text{set}} - T(t)$ as [4], [5]:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt},$$

where $u(t)$ is the control output (heater command) at time t, T_{set} is the desired temperature setpoint, $T(t)$ is the measured temperature at time t, and K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively. The proportional term provides immediate corrective action proportional to the error, the integral term eliminates steady-state offset by accumulating past error, and the derivative term damps overshoot by responding to the rate of change. Together, they offer a practical balance of responsiveness, accuracy, and stability for thermal regulation. In practice, PID gains can be found using the Ziegler–Nichols tuning method, in which the proportional gain is increased until sustained oscillation is observed to determine the ultimate gain K_u and oscillation period P_u, after which empirical rules are used to estimate K_p, K_i, and K_d [6].

PID control was chosen as it is the most apt for controlling single-input single-output thermal systems without requiring the exact model, as well as PID control's proven robustness and widespread adoption in cryogenic instrumentation [4]. While other thermal control strategies such as adaptive control or model-predictive control can handle nonlinearities and time-varying dynamics, they require a model of the system, which is not aligned with our goal of a model-agnostic, generalizable control platform. For the current testbed, where the thermal dynamics are approximately linear near the operating point and the primary goal is demonstrating a functional open-source control platform, PID provides a practical and sufficient baseline. Future work may explore alternative control schemes.

III. Existing Control Platforms

Existing cryogenic control platforms demonstrate that remote monitoring, device integration, and closed-loop operation are achievable, but they often rely on proprietary software stacks or hardware-specific architectures. For example, CADS uses embedded HTTP-based device servers with centralized LabVIEW® supervision, while the NICA test bench adopts Tango Controls for thermometry, refrigeration control, archiving, and remote access [7], [8].

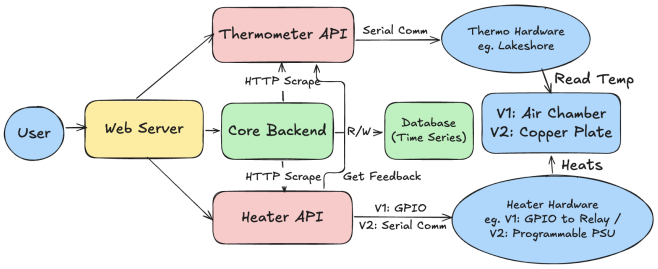

Fig. 1. Overview of the proposed software architecture.

This project provides three benefits beyond those reported in existing systems:

- **Remote web-based operation:** The system supports remote monitoring and control through a web interface, to adjust PID parameters and setpoints without on-site presence.
- **Fully open software implementation:** The software stack is implemented with no proprietary software dependency; instrument communication relies only on the Lake Shore library available in the environment.
- **Generalizable laboratory abstraction:** The Version 2 setup represents a generic real-world thermal-control scenario, where a conductive copper medium links a cooling source (Peltier modules in this implementation) with counteracting PID-controlled polyimide heaters.

IV. Software Architecture

Fig. 1 summarizes the proposed software stack. The design follows a layered architecture that separates hardware interfacing from application logic to keep components modular, independently deployable, and easy to extend. Hardware devices are wrapped by lightweight adapter services that expose a uniform REST interface [9], [10]. System services then orchestrate control, data collection, and visualization on top of these adapters.

A. Hardware Services

Hardware services interface directly with laboratory devices and expose measurement and control functions over HTTP via a REST interface. These services consist of:

1) Thermometer API: This API runs on a host connected to the temperature sensor (currently optimized for Lake Shore devices) and exposes the capabilities provided by the official SDK through a REST interface for remote readout and configuration.

2) Heater API: The Heater API provides remote control of the heater actuator (e.g., via GPIO pins or a programmable power supply interface). It also executes the PID loop by continuously querying temperature feedback (via the Thermometer API) and applying the corresponding heater output. The service exposes both configuration endpoints (e.g., PID parameters, setpoints) and runtime metrics (e.g., current temperature, output power, PID state).

B. System Services

System services implement the core application logic without directly interfacing with hardware, allowing them to be deployed independently from the physical lab equipment.

1) Web Application: Serving as the user-facing dashboard, the Web Application employs a *backend-for-frontend* pattern to provide real-time status monitoring, historical data plots, and configuration pages for the connected devices and control loops.

2) Core Backend: The Core Backend provides three primary functions:

- **System registry:** Maintains a persistent registry of the configured physical rigs, each described by a Thermometer API and Heater API endpoint, allowing a single dashboard instance to supervise multiple testbeds concurrently without code changes or reconfiguration.
- **Metrics collection:** Periodically scrapes measurements and controller states (e.g., temperature, heater output, PID status) from the registered hardware services and stores them in a TimescaleDB time-series hypertable.
- **Query service:** Serves time-bucketed historical data to the web dashboard for dynamic plotting and post-experiment review.

C. Architectural Coupling Limitations

The current architecture exhibits a hardware coupling limitation: the Thermometer API is tightly integrated with Lake Shore instruments, while the Heater API is designed specifically for the Raspberry Pi's GPIO or a particular programmable power supply model. This hardware dependency restricts the platform's adaptability for researchers utilizing different equipment.

To address this limitation, future work will define standardized API contracts that allow users to implement hardware-agnostic thermometer and heater services, enabling seamless integration with our core modules regardless of the underlying hardware.

D. Local Development Tools

To support development without continuous access to the physical setup, we implemented an environment simulator that acts as a drop-in stub for the hardware services. The simulator uses a simple linear thermal model, not intended to replicate a real environment but enough for its purpose of validating the *software integration flow*: that all API services can communicate with one another, that control logic issues commands and receives responses correctly, and that the web dashboard reads and displays values as expected. This enables hardware-free development and rapid feature validation (e.g., dashboard changes or new API endpoints) without requiring access to the physical laboratory setup. An automated end-to-end test suite further launches the entire containerized software stack against this simulator on every code change, catching integration regressions before deployment.

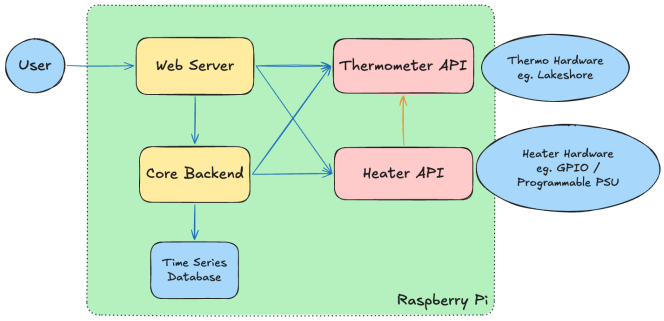

Fig. 2. Deployment Configuration A: Single-machine setup.

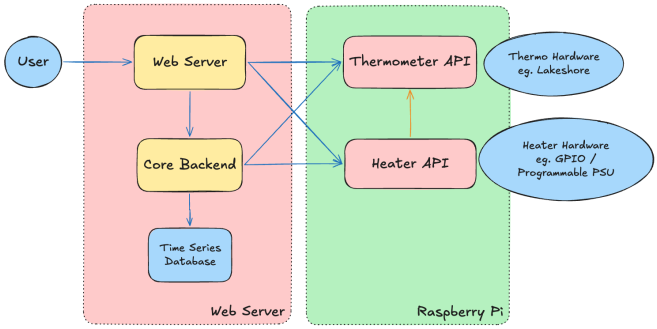

Fig. 3. Deployment Configuration B: Distributed multi-machine setup.

E. Deployment

Although all services are designed to be independently deployable, certain inter-service communications require low latency. These latency-sensitive connections are illustrated by the solid orange lines in Fig. 2 and Fig. 3.

Two deployment configurations are evaluated. In Configuration A (Fig. 2), all services are deployed on a single physical machine, providing a straightforward setup and minimizing network overhead. In contrast, Configuration B (Fig. 3) separates the System Services and Hardware Services across different machines. This separation is feasible because the communication between these specific layers (indicated by the blue lines) is not strictly time-sensitive, and can tolerate delays on the order of several hundred milliseconds.

In the current setup, we use Configuration A (Fig. 2) due to its simplicity in the early stage of development. In the future with a more complex system, Configuration B (Fig. 3) would be more feasible as this approach enables greater flexibility in resource allocation, improves fault isolation, and allows independent scaling or maintenance of the respective services.

With the exception of the Web Application, all services are accessible only within the internal network. Consequently, the Web Application serves as the sole, secure entry point for remote monitoring and external control.

All first-party services are packaged as OCI container images and deployed via Docker Compose, which simplifies the deployment process and ensures strict isolation of software dependencies across heterogeneous runtimes.

F. Monitoring and Observability

We employ a comprehensive observability stack consisting of logs (*Loki*) [11], traces (*Tempo*) [12], and metrics (*Prometheus*) [13], integrated via OpenTelemetry [14] for instrumentation and data collection. This setup enables the systematic monitoring of system behavior across multiple dimensions [15].

A primary metric of interest is the end-to-end latency between the Heater API and the Thermometer API, as the effectiveness of PID control is highly sensitive to delays in both feedback and actuation. Excessive latency or variability can degrade control stability and responsiveness.

By continuously collecting and analyzing latency statistics, tracing spans, and runtime metrics, we are able to quantify system performance, identify communication bottlenecks, and assess their direct impact on control behavior. These empirical insights inform ongoing design decisions and guide future architectural improvements.

The source code for this project is available on GitHub at https://github.com/Q2TM/low-temperature-control.

V. Hardware Configuration

This section presents the hardware architecture of the thermal proxy platform, which evolved through two development phases: Version 1 (insulated chamber prototype) and Version 2 (copper thermal core redesign). This two-phase approach refines the hardware to better approximate qualitative cryogenic thermal dynamics within a laboratory-scale testbed.

A. Version 1: Insulated Chamber Prototype

1) System Overview: Version 1 implements a thermal control system using an insulated chamber as the controlled environment to validate temperature sensing, actuator integration, and PID-based closed-loop control. The insulated enclosure reduces heat exchange with the external environment and increases the thermal time constant, enabling stable PID tuning.

2) Hardware Components: Temperature measurement is performed using the Lake Shore Model 240 temperature monitor [16], which provides high-precision resistance-to-temperature conversion and communicates with a Raspberry Pi 5 controller via USB. Polyimide resistive heater modules (approximately 25 W each) are installed inside the chamber and controlled through a relay-based GPIO switching circuit; two modules provide sufficient dynamic range for PID response characterization. Four thermoelectric Peltier modules provide active cooling for bidirectional temperature regulation. The Raspberry Pi 5 serves as the central controller, managing sensor communication, heater actuation, PID execution, backend API hosting, and data logging.

3) Version 1 Characteristics: Version 1 exhibits high thermal inertia and slower transient response due to convection-dominated heat distribution, resulting in stable, predictable PID behavior. The complete hardware setup is illustrated in Fig. 4.

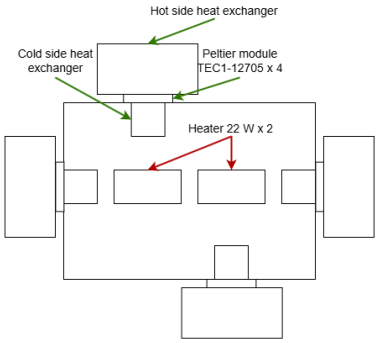

Fig. 4. Insulated Chamber Prototype Diagram.

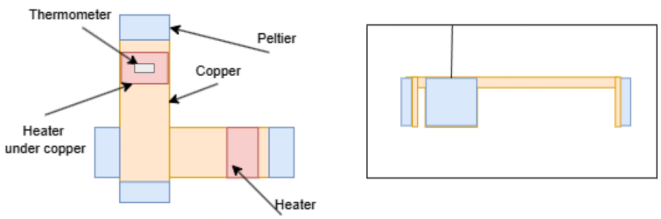

Fig. 5. Copper Thermal Core Structure.

B. Version 2: Copper Thermal Core within Insulated Chamber

1) Copper Thermal Core Design: Version 2 enhances the internal thermal structure while preserving the external insulated enclosure. A pair of 3 mm thick copper bars are mounted inside the chamber as a conductive thermal core, selected for its high thermal conductivity and uniform heat spreading capability. Unlike Version 1, where heat distribution depends primarily on air convection, Version 2 achieves conduction-dominated heat transfer, yielding reduced internal temperature gradients and faster heat propagation.

2) Actuation and Control Upgrades: Resistive heaters are directly attached to the copper bars to minimize thermal interface resistance. Heater power delivery is driven by a programmable power supply (PSU) via a standard serial interface, replacing the GPIO-PWM relay scheme of Version 1. This provides smoother output regulation and a more direct mapping between control commands and thermal input, improving PID tuning sensitivity and experimental repeatability. Peltier modules are also mounted to the copper structure for efficient heat extraction.

These upgrades are expected to yield a lower thermal time constant, higher spatial temperature uniformity, and improved control bandwidth compared to Version 1, transitioning the platform from a validation-oriented prototype toward a more performance-optimized thermal control system, as shown in Fig. 5.

C. Safety Mechanisms

Since the testbed involves resistive heating with remote operation, several fail-safe mechanisms are integrated into the Heater API. A software-defined maximum temperature

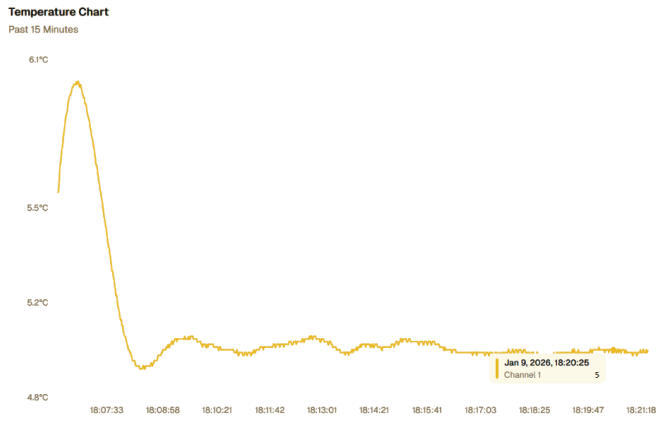

Fig. 6. Version 1 PID result: measured temperature versus time.

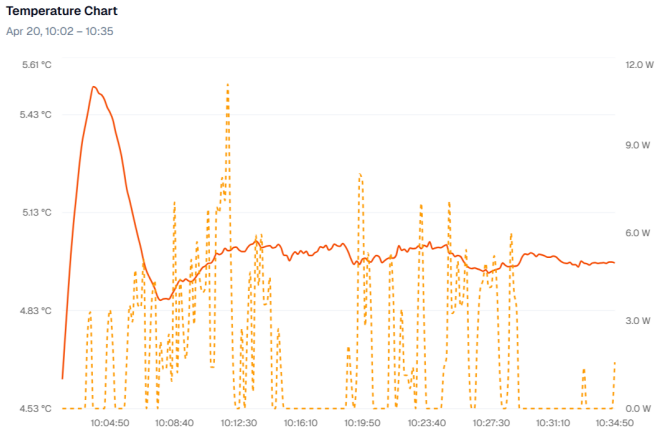

Fig. 7. Version 2 PID result. Red solid line: measured temperature (left axis, °C). Orange dashed line: commanded heater output power (right axis, W).

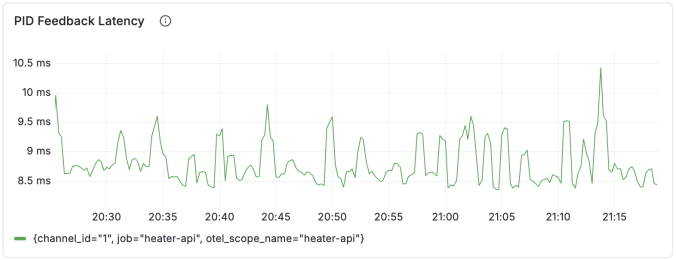

Fig. 8. Measured latency between Heater API and Thermometer API during a PID loop.

threshold immediately disables heater output if exceeded, and the heater is turned off upon critical system errors. A watchdog timer monitors communication between the Heater API and the Thermometer API—if no valid reading is received within a configurable timeout, heater power is automatically shut off.

VI. Experiment Results

Fig. 6 shows the closed-loop temperature response for Version 1, plotted as a single temperature trace, achieving setpoint tracking at $5\,^{\circ}\mathrm{C}$ with limited overshoot and low steady-state error. The PID gains ($K_p = 60$, $K_i = 0.9$, $K_d = 1012.5$) were obtained using the Ziegler–Nichols closed-loop tuning method. The ultimate gain K_u and oscillation period P_u were identified by increasing the proportional gain until sustained oscillations occurred with K_i and K_d disabled. Initial parameters were then calculated using standard Ziegler–Nichols rules and subsequently fine-tuned to improve stability and reduce overshoot under the system's operating conditions.

As for Version 2, Fig. 7 shows two traces on dual axes: the red solid line is the measured temperature (left axis, °C) and the orange dashed line is the commanded heater output power (right axis, W), allowing the control effort to be visualized alongside the thermal response. Version 2 similarly achieved setpoint tracking at $5\,^{\circ}\mathrm{C}$ with even lower overshoot than Version 1, improving from $6.02\,^{\circ}\mathrm{C}$ to $5.52\,^{\circ}\mathrm{C}$ peak, while also showing low steady-state error; using Ziegler–Nichols, the PID gains were ($K_p = 40$, $K_i = 0.615$, $K_d = 1733.33$).

Both versions of the implemented PID controller achieve stable closed-loop regulation: the system reaches the desired setpoint with controlled rise characteristics, bounded overshoot, and small steady-state error. After convergence, temperature fluctuations remain within a narrow band around the target value.

Fig. 8 shows the feedback communication latency on Version 2 with deployment Configuration A, as described in Section IV-E. This measurement was enabled by the observability infrastructure introduced in Section IV-F, which was not yet in place during Version 1 development. The latency is typically around 8.4–10.4 ms. The signal remains bounded without long-term drift, indicating predictable timing for closed-loop execution. This latency is much smaller than the dominant thermal time constants and is therefore not a performance bottleneck at this stage.

The experimental results also confirm reliable communication between hardware and controller, stable real-time PID execution tolerant to random communication errors, and effective actuation through the programmable power supply serial interface. The integration of API-based parameter control further validates the system's usability and modularity.

A. Hardware Evolution: Version 1 to Version 2

Version 1 served as the integration baseline with relay-based heater switching. While it demonstrated stable PID behavior, it exhibited high thermal inertia, slower transient response, and moderate internal temperature uniformity due to convection-dominated heat distribution.

Version 2 addresses previous limitations by introducing a conductive copper core and replacing the relay with a programmable power supply (PSU). The copper core enables faster, conduction-dominated heat transfer with reduced temperature gradients, while the PSU provides smoother output regulation and direct thermal actuation. These upgrades transition the platform toward a performance-optimized testbed, expecting to yield a lower thermal time constant, higher spatial uniformity, and improved control bandwidth. Although

initial PID gains are tuned via the Ziegler–Nichols method, full closed-loop validation remains pending due to time constraints.

VII. Conclusion

In this paper, we presented the development of an open-source, laboratory-scale thermal proxy testbed designed to emulate qualitative aspects of cryogenic control dynamics—such as long thermal time constants, measurement delays, and counteracting heat loads—at accessible, above-ambient temperatures rather than at true cryogenic conditions. The platform combines a Raspberry Pi 5 controller, a Lake Shore Model 240 temperature monitor, and resistive heating with an API-driven control stack for remote operation and observability. A key hardware evolution in this work is the transition from relay-based heater switching in Version 1 to programmable-power-supply (PSU) current control in Version 2. This transition improves actuation smoothness, provides a more direct command-to-thermal-input relationship, and better supports sensitive PID tuning.

The software architecture decouples hardware adapters from core control services and includes an environment simulator for hardware-free development and software-in-the-loop testing. Experimental results on the insulated chamber prototype show stable closed-loop setpoint tracking with bounded overshoot, small steady-state error, and acceptable API-loop latency for the current thermal time scale. With the Version 2 copper thermal core and PSU-based actuation, the testbed advances from baseline integration validation toward a more precise and scalable platform for education, rapid prototyping, and future transfer to more demanding cryogenic infrastructure in superconducting-qubit laboratories.

VIII. Next Steps

Next steps focus on completing Version 2 closed-loop validation, improving hardware-agnostic APIs and observability, and assessing long-running stability before extending the platform toward sub-zero operation.

Acknowledgment

This work was made possible by advisement and equipment support from Dr. Patryk Gumann, Head of Novel Technologies for Future Quantum Computing Systems from IBM Quantum. We also gratefully acknowledge Prof. Pitchaya Sitthi-amorn for recommending and providing the programmable power supply used in the Version 2 platform. Additionally, we would like to thank Jakkapat Tharaworn from the Department of Electrical Engineering for his assistance with hardware assembly and troubleshooting during the experiments.

References

[1] S. Krinner, S. Storz, P. Kurpiers, P. Magnard, J. Heinsoo, R. Keller, J. Lütolf, C. Eichler, and A. Wallraff, "Engineering cryogenic setups for 100-qubit scale superconducting circuit systems," *EPJ Quantum Technology*, vol. 6, no. 1, p. 2, 2019. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-019-0072-0

[2] SpinQuantum, "Dilution refrigerator: Everything you need to know," 2025, accessed 2026-02-16. [Online]. Available: https://www.spinquanta.com/news-detail/the-complete-guide-to-dilution-refrigerators

[3] Insight i|ħ/Qubit Perspectives, "Hardware guide - dilution refrigerators," 2025, accessed 2026-02-16. [Online]. Available: https://www.insight-ihbar.com/QC_hardware_en/F.html

[4] D. J. M. Swartz and L. G. Rubin, "Fundamentals for usage of cryogenic temperature controllers," accessed: 2026-02-16. [Online]. Available: https://www.lakeshore.com/docs/default-source/product-downloads/literature/3300_fundamentals.pdf

[5] P. Woolf, "9.6: Pid downsides and solutions," Engineering LibreTexts (Woolf, *Chemical Process Dynamics & Control*), 2024, accessed: 2026-02-16. [Online]. Available: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.06%3A_PID_Downsides_and_Solutions

[6] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transactions of the ASME*, vol. 64, no. 8, pp. 759–765, 1942.

[7] J. Antony, D. S. Mathuria, T. S. Datta, and T. Maity, "Development of intelligent instruments with embedded http servers for control and data acquisition in a cryogenic setup—the hardware, firmware, and software implementation," *Review of Scientific Instruments*, vol. 86, no. 12, p. 125003, 2015.

[8] G. Sedykha, E. Gorbache, A. Kirichenko, V. Volkov, A. Galimov, D. Nikiforov, D. Neapolitanskiy, V. Kosachev, and R. Pivin, "Control system of the superconducting magnet test bench for the nica accelerator complex," in *Proceedings of the XXVI International Symposium on Nuclear Electronics & Computing (NEC'2017)*, Becici, Budva, Montenegro, Sep. 2017, september 25–29, 2017.

[9] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[10] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *2010 Internet of Things (IOT)*. IEEE, 2010, pp. 1–8.

[11] Grafana Labs, "Loki: Like prometheus, but for logs," 2026, accessed: 2026-03-26. [Online]. Available: https://grafana.com/oss/loki/

[12] ——, "Tempo: A highly scalable, distributed tracing backend," 2026, accessed: 2026-03-26. [Online]. Available: https://grafana.com/oss/tempo/

[13] Prometheus Authors, "Prometheus: An open-source monitoring system and time series database," 2026, accessed: 2026-03-26. [Online]. Available: https://prometheus.io/

[14] OpenTelemetry Authors, "Opentelemetry: High-quality, ubiquitous, and portable telemetry," 2026, accessed: 2026-03-26. [Online]. Available: https://opentelemetry.io/

[15] C. Sridharan, *Distributed Systems Observability: A Guide to Building Robust Systems*. O'Reilly Media, Inc., 2018.

[16] Lake Shore Cryotronics, Inc., *240 Series input modules*, Lake Shore Cryotronics, Inc., Westerville, OH, USA, 2021, accessed: 2026-03-26. [Online]. Available: https://www.lakeshore.com/products/categories/overview/temperature-products/cryogenic-temperature-modules/240-series-input-modules