Telemetry for Quantum Systems in HPC Centers

Hossam Ahmed[*], Burak Mete[*†], Helmut Heller[*], Matthew Tovey[*], Xiaolong Deng[*], Asim Zulfiqar[*], Muhammad Nufail Farooqi[*], Mahmoud Abuzayed[*], Martin Schulz[*†], Laura Schulz[*]

[*]*QCT Department*
Leibniz Supercomputing Centre of the Bavarian Academy of Science and Humanities (LRZ)
Garching b. München, Germany.
[†]*Chair for Computer Architecture and Parallel Systems (CAPS)*
Technical University of Munich (TUM)
Garching b. München, Germany.

***Abstract*—Quantum systems have traditionally operated within highly controlled laboratory environments to minimize internal noise and maximize stability. However, when these systems are integrated into High Performance Computing (HPC) environments and setup for continuous operation, they encounter significantly higher levels of external noise and instability coupled with higher demands. This paper introduces a comprehensive framework to counter the challenges of maintaining the stability of operational quantum computers in such noisy HPC settings. We propose a novel system architecture that monitors and manages the environmental conditions of quantum computing systems using a broad set of sensors, enhancing their robustness and reliability. Our architecture is open and can be extended to other setups. It can serve as a blueprint for other sites as it spans quantum technologies and offers broad coverage. Ultimately, this data will be essential to facilitate advancements in machine learning to drive real-time error mitigation.**

I. Motivation

Current quantum computers often fail to achieve their advertised levels of fidelity due to a range of factors, including, in part, often overlooked environmental conditions. This issue becomes particularly significant when integrating Quantum Computers (QC) with High-Performance Computing (HPC) systems.

Ideally, these systems should be closely co-located, as such hybrid HPCQC setups enable fast and efficient network connectivity between the two [1]–[4], which is helpful in tightly coupled hybrid algorithms. Further, it allows for easier and more efficient operation as a single system with one scheduling domain. However, such placement in HPC centers and around noise-inducing HPC systems presents vastly different environmental requirements compared to the controlled physics lab environments. As a result, treating quantum computers as accelerators housed within HPC environments necessitates further analysis and detailed characterization to ensure better performance.

Characterizing environmental noise and its impact on Quantum Processing Unit (QPU) efficiency presents a significant challenge. In particular, the different time scales of computation vs. environmental sensors, the inability to directly measure inside the QC, and the unknown effects between the environment and the QCs lead to substantial challenges. Current environmental data collection solutions are often too coarse-grained and limited in their data sources to map the qubits' susceptibility to external factors, This limitation is particularly critical given that precise characterization requires continuous monitoring of qubit decoherence times (on the order of milliseconds) during data acquisition. This disconnect limits our ability to establish crucial correlations between environmental parameters and the QC's performancee xhibited through key metrics, particularly when evaluating operational benchmarks such as qubit error rates, readout fidelity, and single- or two-qubit gate fidelities—metrics that directly reflect system reliability under real-world conditions.

To address this challenge, a tightly integrated system is required—one that seamlessly connects environmental data acquisition systems with QPU operations. By correlating the collection of environmental data with quantum experiments, we can analyze anomalies and irregularities in qubit or gate behavior, enabling a more precise characterization of the impact caused by environmental noise.

The process of noise characterization involves designing specific patterns for QPU operations and evaluating how environmental noise distorts their operation results. Through this approach, the effective noise can be classified as either coherent noise which arises from deterministic unitary processes, whose effects are still characterized by unitary operations (e.g., over-rotations in rotation gates), or incoherent noise which is non-unitary and arises from stochastic interaction with the environment. This classification is a fundamental step towards understanding the sources of noise and developing effective mitigation strategies given the environmental setup parameters. By achieving this level of characterization, we ease the way for optimizing QPU performance in real-world conditions, ensuring higher fidelity and robustness in quantum computing applications.

This paper introduces a telemetry project that is currently monitoring over 508 sensor readings at intervals ranging from one to sixty seconds in an HPCQC setup of a superconducting QC system. The project is organized into several components to facilitate the transfer of Internet of Things (IoT) sensor data to a central database, where the data is subsequently processed using machine learning techniques, enabling the characterization of environmental parameters crucial to quantum comput-

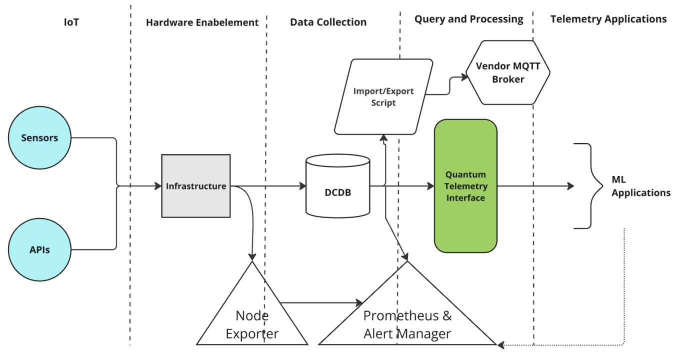


Fig. 1. Overview of the quantum telemetry project.

ing. An overview of the framework of the telemetry project's structure is presented in Fig. 1.

Specifically, this paper makes the following contributions:

- We deduce the requirements and needs for telemetry around quantum systems.
- We detail our comprehensive sensor setup and its technical implementation in a real HPCQC setup at our center.
- We explain the needed network infrastructure in a real-world setting that enables continuous monitoring and analysis.
- We describe and implement the needed software setup to gather, process and analyze the data gathered from our sensors.
- We give a first look into how the data can be used for error mitigation and correction of QC systems.

Overall, this paper presents a practitioner's blueprint of how to setup comprehensive monitoring beyond what is currently available or even feasible in pure HPC environments. With this, it provides a comprehensive overview on how to co-design lab space, QC systems and analysis software with the goal of improving and stabilizing QC systems outside of controlled lab environments.

The remainder of the paper is organized as follows. In Sec. II we motivate the work by showing two case studies in which monitoring was essential to understand system behavior. In Sec. III we introduce our comprehensive IoT setup, followed in Sec. IV by the needed network setup and in Sec. V with the notification mechanisms connecting the sensors to the network. In Sec. VI we discuss the matching software setup, completing the overall solution. In Sec.VII we then show how the data can contribute to stabilizing QC systems, before, in Sec. VIII, concluding the paper with final remarks.

II. Monitoring and Characterizing QC Performance

Characterizing qubit performance is one of the key tasks in the design of any quantum computer. However, it is typically executed when designing and testing the actual QPUs, while being in a controlled physics laboratory environment, and not under production conditions. Additionally, these approaches typically focus on isolating and analyzing a single environmental factor. In contrast, our objective is to capture data from all relevant environmental factors in a production environment. Rather than establishing direct correlations, our goal is to uncover and analyze the underlying relationships and interactions between these factors, recognizing that some correlations may depend on others. This novel approach aims to provide deeper insights into the complex interplay between environmental influences and qubit performance.

Our approach is currently employed at our HPC center where we have three superconducting QPUs and one trapped ion quantum system co-located near one another. In this setup of multiple quantum systems, we deploy and monitor over 500 sensors for different environmental data using a comprehensive IoT setup. Some of these sensors directly record physical room properties (e.g., temperature or humidity), while others record one or a group of specific quantum system parameters available from the QC setups themselves (e.g., stemming from the cryostat controller).

A. *Motivating Case Study: Superconducting Systems*

Advancing quantum computing requires addressing the critical challenge of identifying and characterizing the noise sources that affect quantum systems. These sources are broadly classified into coherent noise [5], which arises from predictable and systematic errors, and incoherent noise, which stems from random, uncorrelated disturbances. The latter is particularly problematic as it destabilizes quantum operations and reduces fidelity.

As one example for incoherent noise, we used our setup to observe changes in the vacuum can pressure of a cryostat, as it is used for superconducting QC systems, which had a direct correlation to qubit fidelity, i.e., quality of the targeted qubit. This led us to examine additional parameters, including the ambient temperature of the room housing the compressors and the gas handling system of the cryostat. Notably, we identified a correlation between room temperature and solar activity (Fig. 2), as indicated by the light intensity sensor. This represents an instance of incoherent noise affecting quantum systems, which must be mitigated to ensure optimal operation.

Characterizing and understanding such noise sources are critical steps in improving the environmental control of quantum systems. Incoherent noise, such as that caused by environmental temperature fluctuations and light intensity variations, disrupts the quantum states by introducing random, uncorrelated errors that accumulate over time. By identifying the coherence properties of these disturbances, we can implement strategies to isolate and counteract their effects. For example, we started implementing solutions for controlling the temperature also in the room that is housing the gas handling systems (which is separate from the room housing the QC system itself) to mitigate this incoherent noise. These efforts enable us to achieve higher fidelity and more stable performance in quantum computing applications, underscoring the importance of thorough environmental noise analysis.

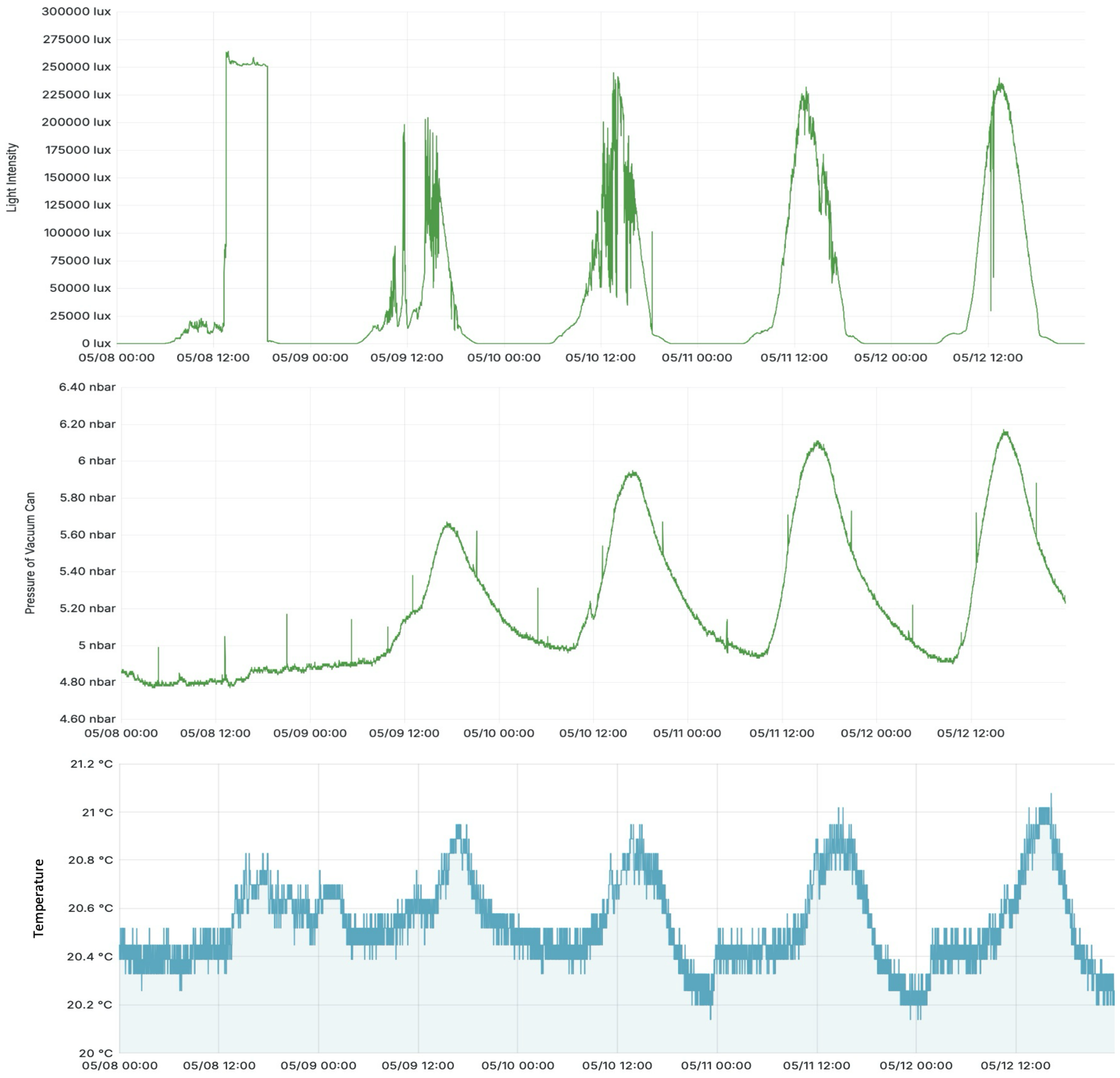
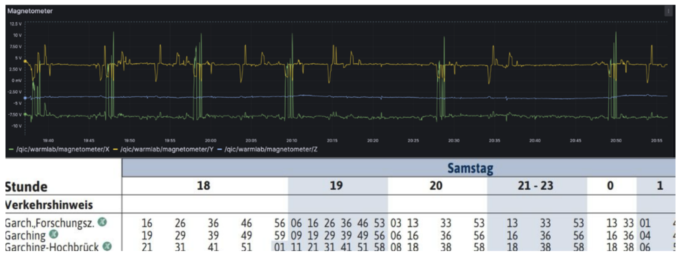


Fig. 2. visual representation for data within specific time frame between the changes of vacuum can pressure, light intensity and temperature showing the direct correlation between them.

B. Motivating Case Study: Trapped-Ion Systems

However, in quantum systems, noise is rarely purely coherent or incoherent; instead, it often manifests as a combination of both. Coherent noise, which stems from systematic and predictable disturbances, can be analyzed and mitigated through careful design and sensor integration. For instance, we investigated electromagnetic field interference as a form of coherent noise in a Trapped Ion quantum computer. In this study, we placed an electromagnetic sensor near the lasers responsible for trapping the ions. The electromagnetic sensor allowed us to detect specific noise effects, such as state detection errors correlated with the Y-axis of the magnetic field sensor or phase shifts corresponding to the X-axis of the laser sensor. These correlations provided insights into how the electromagnetic field interacts with quantum states, enabling targeted strategies to minimize the interference, such as the strategy discussed in Sec. VII.

Identifying and mitigating coherent noise is critical for maintaining the stability and accuracy of quantum operations. One observed example of what is considered a coherent noise



	Samstag					
Stunde	18	19	20	21 - 23	0	1
Verkehrshinweis						
Garch.,Forschungsz.	16 26 36 46 56	06 16 26 36 46 53	03 13 33 53	13 33 53	13 33	01
Garching	19 29 39 49 59	09 19 29 39 49 56	06 16 36 56	16 36 56	16 36	04
Garching-Hochbrück	21 31 41 51	01 11 21 31 41 51 58	08 18 38 58	18 38 58	18 38	06

Fig. 3. Correlation between electromagnetic field intensity and train schedule. The electromagnetic sensor readings show a distinct pattern corresponding to train arrivals every 10 minutes compared to train arrivals every 20 minutes, highlighting the impact of train activity on the electromagnetic field intensity.

is the electromagnetic noise induced by subway trains passing near our lab, which we observed by comparing the schedule of the train and the occurrence of electromagnetic noise, as shown in Fig. 3. Since we can predict the timing and intensity of this noise, it is possible to mitigate its effects by analyzing how the trapped ion system responds to these disturbances. By applying corrective changes at specific timestamps aligned with the train schedule, the impact of this noise on the system can be effectively mitigated.

III. Internet Of Things for Telemetry

Investigating the effects of environmental changes on quantum computation is an active field of research [6] and it not clear, yet, which environmental quantity will have an influence. As a consequence, we designed a system setup that can collect a wide variety of physical quantities, some of which may and some of which may not influence the behavior of our QC. We refer to our installed sensor setup and its supporting systems that enable continuous monitoring, collection and analysis as ”telemetry”, since it gives us remote insight into the QC and its environment, similar to other telemetry setups, e.g., in automotive, aviation, or space applications. This will ensure that we get a complete overview of all possible factors, so that our analysis can determine which effects are impactful and which are not. Our goal hereby is to a) detect possible impacts onto the performance of our own quantum systems, and b) help co-design QC systems by suggesting key mitigation needs, and c) support future HPCQC setups by guiding their monitoring requirements.

The installation of such a plethora of sensors is a long process that includes sensor selection, ordering, delivering, installation, configuration, and finally data transmission and analysis. Further, the set of sensors has to evolve as we learn more about the system's behavior and as more sensor types become available. Hence, the setup of our sensor suite is an ongoing project, and we will add more sensors to it over time, as needed. However, the current setup (described below in Sec. III-B) already reflects a very wide range of sensors that capture—to our current knowledge—all relevant elements. It is currently employed in our compute center, which houses a 5 qubit and a 20 qubit superconducting QPU, a 20 qubit ion-trap system, as well as an HPC system in close proximity. In our setup, the sensors are located close to the cryostat of the superconducting system, the control electronics of both systems, and the gas handling system (GHS) for the croystat. As additional QC systems are added to our set-up, additional sensors will be placed in close proximity to them to assess environmental quantities close to the respective QCs.

A. Telemetry Embedded Control System

A Raspberry Pi 4B (RPI) is running as the central IoT gateway. It runs Raspberry Pi OS lite 64-bit (for RPI-4), Debian GNU/Linux 11 (bullseye). The sensors are connected to the RPI and the RPI sends the data to a central database for collection, storage and analysis. For this we use the openly available Data Center Data Base (DCDB) [7]. To monitor its health, it is running a Prometheus node-exporter [8], which is monitored by a central Prometheus instance, as outlined in Sec. V.

The RPI and its associated platform provides flexibility to integrate a wide range of sensors of different types and connectivity. For example, the 1-wire bus for our temperature sensors (of type DS18B20) was connected to the serial port of the RPI, while our I2C-capable sensors are directly connected to the RPI's I2C bus. Further, the microphone's ADC is connected to the USB of the RPI.

Where possible, measurements are triggered by CRON in regular intervals. The respective interval depends on the sensor and its needed granularity. For example, this is done

- every minute for the 1-wire temperature sensors,
- every 5 minutes for the liquid nitrogen scales, and
- every 30 minutes for the heat meters.

Using CRON has the advantage that the program is started over and over again and thus can recover from temporary crashes.

However, some programs must run continuously and they are run as daemons. These are:

- Movement detector via a Passive InfraRed (PIR) sensor, triggered by an interrupt whenever the PIR changes its output state. To avoid triggering by spurious signals, each trigger event is verified by reading the pin status shortly after the interrupt. An event is only recorded if the pin is read to be in the active state.
- Doors and windows contacts. Here, the same logic as for the PIR sensor is used.
- Magnetometer data are constantly read with maximum speed, resulting in a measurement every two seconds.
- Light intensity, air pressure, air humidity, and air temperature are read every minute via I2C.
- The loudness sensor is read every second.

The current status is described for each sensor in the following sections.

B. Environmental Sensors

After the general setup description, we now detail all currently available sensors installed in the quantum lab for the telemetry project.

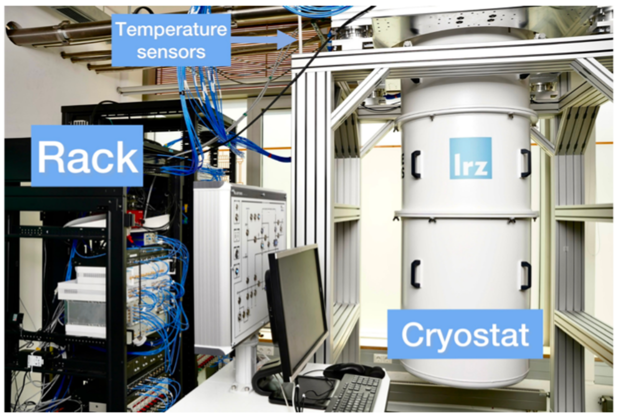


Fig. 4. Superconducting quantum computer column with 35 temperature sensors temperature column.

*a) **Temperature**:* While the main purpose of the croystat of a superconducting QC is to isolate its contents from the environment, some coupling is still present, as evidenced by the rising internal temperature once the active room cooling has been switched off. Keep in mind that the same quantum devices used for QC are also used in quantum sensors to measure the smallest quantities.

To assess the temperature fluctuations outside the cryostat, we install 35 temperature sensors of type DS18B20 from Analog Devices (formerly Maxim Electronics, formerly Dallas Semiconductor) in various heights ranging from 0 cm to 340 cm above floor height in increments of 10 cm. This allows us to quantify the temperature fluctuations in the immediate vicinity of the fridge in all relevant heights of the room.

The DS18B20 sensors are all connected via the so-called 1-wire bus to our IoT gateway in our lab, a Raspberry Pi computer. The connection between the 1-Wire devices and the RPI is made via a serial port to the 1-Wire interface chip, the DS2480B. We employ this specialized chip instead of bit-banging the 1-Wire bus with an IO pin of the RPI because it allows for much better control of the signals on the bus and can thus address a much greater number of devices in a much larger network.

A second temperature sensor of type DS18B20 is placed in the adjacent machine room to monitor the GHS environment. The physical distance is ca. 15 m, which is no problem for the 1-Wire bus.

A third temperature sensor of type BME280 from Bosch is placed at the other end of the room on top of a computing rack to assess the temperature variations within the room. It is connected via an I2C bus to the Raspberry Pi.

The temperature in the room is regulated by one or two AC cooling units (if needed) in proximity to the cryostat.

*b) **Air Pressure**:* The BMS280 is a combined temperature, pressure, and humidity sensor and is also used to measure the air pressure in the room. Since the room is connected to the outside (of the building) environment through its air ducts, the pressure in the room varies with the outside pressure and thus with the weather.

*c) **Relative Humidity**:* The relative humidity (rh) is also measured by the BMS280. Proper humidity is important for the proper operation of the QC electronics. It must be within strict limits given by the vendor and cannot be too dry or too wet. Initially, the humidity in the room was not regulated, but only measured. A new humidifier is now installed to keep the humidity level in the room stable regardless of the change in the seasons. Regulation of temperature and humidity is a must, especially in our case, where we have a superconducting quantum computer and an ion trap quantum computer in the same space with very different temperature and humidity requirements.

*d) **Movement**:* The presence of humans next to the QCs might influence them through vibrations or electromagnetic fields, e.g., from their cell phones, which operate in a frequency range comparable to the frequencies used in superconducting QCs. We use a passive infrared (PIR) sensor, type HC-SR501, which is used in millions of outdoor lights to turn them on when people approach. It passively senses the body heat of living beings and closes contact if a *change* of the surrounding heat distribution is detected. The sensor is mounted centrally in the lab room at about 4 m height to give it a good view of most of the room. We tested its sensitivity and it triggers reliably from anywhere in its field of view. Its output pin is directly connected to the RPI.

*e) **Doors**:* The two entrance doors to the lab are fitted with magnetic contacts to indicate whether they are open or closed. Additional mechanical contacts in the door locks indicate whether the door is locked or not. These simple switches are directly connected to GPIO pins on the RPI and to the ground on the other side. A programmable pull-up resistor in the RPI's GPIO network pulls the pin to HIGH when the switch is open.

*f) **Light Intensity**:* Light intensity can give information about whether the fluorescent ceiling lights are switched on or off, whether the sun is shining into the room through the windows, and in general, whether it is dark in the room or not.

The fluorescent ceiling lights are known to emit a wide electromagnetic interference spectrum so knowing whether they are on or off at any given time is interesting. Since we don't have sensors on the lights or switches themselves, we use this indirect method to deduct their state.

Sunshine can increase the temperature of vacuum pipes or the outside of the fridge without really increasing the room temperature itself (since the room is actively cooled and the room temperature is regulated), so it is good to know whether the sun is shining in or not.

We install a sensor of type BH1750 [9] on top of a computing rack, looking upward towards the ceiling, close to the RPI so that the I2C connection is only about one meter long.

*g) **Noise Intensity**:* Sound is pressure vibrations of the air, which, in turn, can couple to the QC through the frame or the cryostat vessel and influence the computations. Therefore, we measure the acoustic noise level in the room, that is the peak values of the sound intensity. In order to avoid complications with data protection issues we only record the low pass filtered noise *level* in the room, but not the direct sound itself, which would allow us to reconstruct conversations held in the lab.

We use the Grove loudness sensor [10]. Since this sensor only delivers an analog output signal and since the RPI has no analog input at all, we need an analog-to-digital converter (ADC). We used the Grove Pi Plus HAT [11], which has three ADCs (and seven digital connections). The Grove Pi Plus uses the I2C bus to connect to the RPI. We record the sound intensity every second and store the data in the DCDB.

h) ***Magnetometer:*** Magnetic fields play a major role in superconducting QCs, in ion trap QCs, and neutral-atom QCs. The magnetic fields involved are very small, so good magnetic shielding of the QCs is important. However, even the best conventional shielding can never exclude external magnetic fields completely, only dampen it. A small coupling of external magnetic fields to the active parts of QCs is unavoidable and since the QCs are very sensitive to even small magnetic fields permeating their active quantum parts (resonators, ions, atoms, etc.), external magnetic fields can influence the fidelity of the QC.

Since already small magnetic fluctuations can influence the QC, we do not use a cheap chip-type sensor, like the BMM 350 [12], which can be bought for under 2€, but only has a resolution of 30 nT. These sensors have become so cheap because every cell phone has one inside as a magnetic compass. Instead, we opted for an industrial magnetometer from Bartington, the MAG-13MS100 [13] with a resolution well into the pico Teslas. This sensor needs a power supply and a special pre-amplifier. We select the SCU-1 [14] from Bartington, which offers also signal amplification up to 1,000 X. This still leaves us with an analog signal that we have to convert to digital. The ADC from the Grove Pi Plus HAT only has a 10-bit resolution, which is not good enough for our usage, so we decide to use a separate high-resolution ADC for the magnetometer. We use a HAT from Waveshare [15] with a phenomenal 32-bit resolution, allowing us to see the smallest fluctuations down in the electric noise. The board uses an ADS1263 chip [16] from Texas Instruments.

i) ***Electrical Power Monitoring:*** Energy consumption is a key performance metric in today's HPC environments. Existing supercomputers consume power in the megawatt (MW) range, such as Frontier‘s 23 MWs and Fugaku‘s 30 MWs. Compared to these supercomputers, quantum computers (QCs) consume power in the kilowatt (KW) range. For instance, superconducting QCs operate at extremely low temperatures, while neutral atom QCs and trapped ion QCs operate in ultra-high vacuums, both of which require special conditions to maintain. We briefly discuss the main core components of QCs and their power consumption. The core components of superconducting include dilution refrigerators and classical control systems, with a power consumption of approximately 25 KW and 2 KW, respectively, as shown by our setup. The core components of trapped ion QCs include ion traps, lasers, vacuum systems, and control electronics, with power consumption estimated to be around 2 KW. Similar to trapped ion QCs, neutral atom QCs have optical lattices and tweezers, laser cooling, vacuum generators, and control electronics. Their power consumption is also estimated to be approximately 2-3 KW. In our lab, these three different types of quantum systems have been deployed or are being deployed. The core and other components are connected through their respective PDUs to the power supplier and their energy consumption is monitored in real-time and recorded into our DCDB system. All of the data and graphs can be checked and seen via Grafana, as well. We also monitor the power consumption of the nearby elevators and all the servers connected to our quantum computers as well as the controller electronics in order to notice any changing behavior or power consumption in relation to events on the machines.

j) ***Air Conditioning Heat Meters:*** The heat meters are intended to measure the amount of heat coming from the GHS and its compressors, and also the amount of heat from the two air conditioning units (AC) in the laboratory space. They measure the temperatures of the inflowing and outflowing cooling water as well as the amount of water flowing and compute the amount of heat transferred to the cooling water.

We use Sontex Supercal S5 heat meters with the MODBUS module. RS485 connections are then used to connect MODBUS to the RPI, which then stores the values in DCDB.

k) ***Sound in Machine Room:*** During the operation of the cryostat, it turned out that the GHS and the compressors in the machine room emitted a characteristic noise envelope. We pick up this sound with a Thomann microphone of type Behringer SL85S. It is mounted on a tripod and can easily be positioned in the room. It has an XLR connector and the analog sound signal is routed to the RPI via a shielded audio cable. The analog-to-digital conversion is done in a professional ADC converter, also from Thomann, type t.bone MicPlug USB (XLR to USB). The digitized sound signal is then available on a sound device on the RPI for further analysis or transmission to DCDB. Since we have a pipeline that captures audio input on the Raspberry Pi (RPI) and routes it through a USB audio interface, the approach might start simply with FFT-based analysis and thresholding: establish a baseline spectrum, measure deviation from it, and raise alerts if a certain frequency band spikes above normal. Over time, we can refine this by implementing more sophisticated methods to reliably detect unusual frequency components in the lab's hardware noise signature.

l) ***Liquid Nitrogen Supply:*** To cool the dirt traps on our cryostats we use liquid nitrogen in Dewars, which are open to the atmosphere, thus nitrogen is constantly evaporating and the Dewars need to be replenished from time to time. A third, bigger Dewar is used as a reservoir to fill the smaller ones when needed. To know when we have to refill either Dewar, we put them on electronic scales (two from Kern, type IFB 60K-3L-2023e - TIFB 60K-3L-B; the bigger one

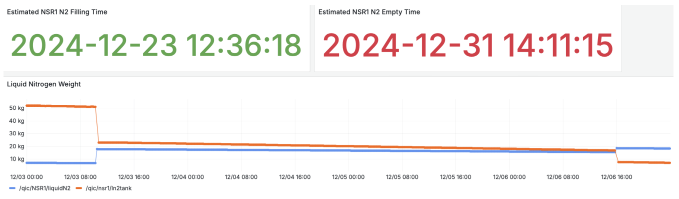


Fig. 5. Liquid nitrogen refilling and evaporation. The blue line and orange line show the dispersion rate at two different locations.

from Henk Maas Wegschalen B.V., type NH8486-400kg), interrogate them every 30 minutes from the RPI and write the weight minus the respective empty weight to our DCDB instance. The evaporation is quite regular, so we calculate the projected date when a refill is needed and send out alarms to our operators to order new nitrogen from our supplier when the reservoir is running empty. Fig. 5 shows two refilling events on December 3rd and December 6th. This also shows how the dispersion rate of liquid nitrogen varies in different locations.

*m) **Dust**:* Dust particles can affect the performance of different quantum computing technologies. Whether warming up the cryostat for maintenance or affecting the performance of the tweezers in Ion-traps and neutral atoms quantum computing systems, dust particles cause damage to quantum systems. We, therefore, also check for dust particle count and size around the QPU to avoid potential damage.

*n) **Cryogenic Measurements**:* The data from the cryostat sensors are collected through a rest API to which we configure our DCDB instance to get the most important sensor readings that are important for the health of the cryostat and through observations to the correlation with superconducting QPU fidelity, as also observed by [6]. The monitored cryostat pressure metrics are: **1) P1**, Vacuum can pressure, **2) P2**, still line pressure, **3) P3**, condensing pressure which is the pressure of the gas inside the condensing line. **4) P4**, Back pressure **5) P5**, Mixture tank pressure **6) P6**, Service line manifold pressure which is used for pumping and cleaning the traps, or recharging the tank. This pressure is very important to monitor during maintenance.

Further, we record **flow** of the circulation of the helium mixture inside the cryostat. and the temperature at different flanges in the cryostat such as:

MC-Temperature, Mixing chamber temperature is the lowest temperature in the cryostat (around 8mK) where the QPU is located. **Still-Temperature**, is the temperature at the still flange (1K Kelvin). **4K-Temperature**, the temperature at the 4 Kelvin flange. **50K-Temperature**, the temperature at the 50 Kelvin flange.

C. Data Transfer to DCDB

Data are sent to our DCDB instance via the MQTT protocol. Since DCDB can only store integers, float values must be multiplied by a large enough number so that they can be truncated to integers without loss of accuracy [7].

a) Data Exchange with Outside Vendors: Access to our Data Center Database (DCDB) instance is restricted to the internal network, limiting direct data exchange with external groups. To address this limitation, we adopt an indirect approach for data sharing. A device provided by the outside vendor, comprising an STM32 microcontroller, temperature and humidity sensors, an accelerometer, and a magnetometer, is installed in our lab. This device publishes its data to an MQTT broker. We have access to all vendor MQTT brokers (of all QC systems in our lab) and have developed two Python scripts for data import and export. These scripts run as user-defined system services to ensure continuous operation.

The import script connects to the vendor MQTT brokers using the Python Paho MQTT client. It subscribes to all device topics using the wildcard *STM32/#*. When a message is received on a subscribed topic, the script decodes the payload, appends a current timestamp, and stores the value in the DCDB under the topic */planqc/qic/warmlab/*.

The export script also connects to the vendor MQTT brokers using the Python Paho MQTT client. It retrieves data from the DCDB and publishes it to the corresponding MQTT broker topics. The messages are formatted in JSON for consistency. The data exported includes various sensor readings such as magnetometer, humidity, temperature, pressure, light, loudness, movement, dust, door lock status, and power usage metrics.

D. System Capacity

Currently data are being read from about 500 sensors for QC systems at intervals of one second to one hour, which is well below the capability of telemetry systems that can process 100,000 sensor readings per second [7]. Furthermore, for real-time analysis of environmental effects on quantum systems, only specific readings that are highly correlated with the target quantum system (superconducting or trapped ions) are used. However, if required, the system can be scaled horizontally to handle higher loads.

IV. Network Infrastructure

Virtual Extensible LAN technology (VxLAN) is being used in order to be able to provide a particular VLAN on a particular port of any switch in the infrastructure. We make heavy use of multiple VLANs, either routed or unrouted, in the lab. Routed VLANs are assigned a particular subnet and a gateway address will be supplied through which the rest of our instititional backbone network Munchner Wissenschaftsnetz (MWN) [17] can be reached. Unrouted VLANs build a private L2 network that is only accessible via whatever systems are connected to those ports.

Most subnets in the infrastructure sit behind a NAT device for security, so private network addresses are used most of the time.

A. Layout example

As an example of our network layout, Fig. 6 shows the current configuration of a switch in our lab. The separate VLANs are listed in Table I. Of note for this article are the following points.

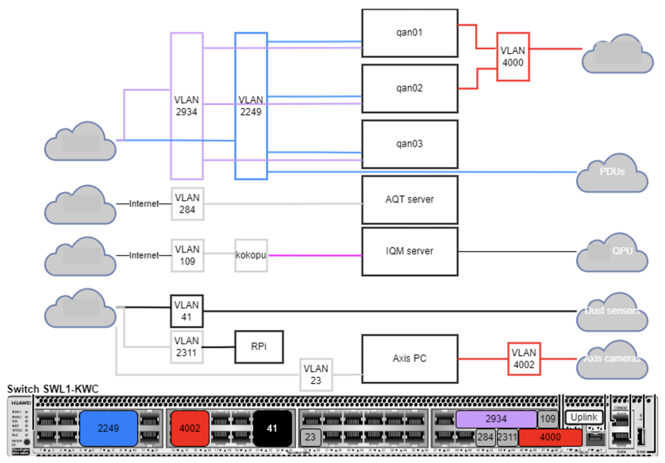


Fig. 6. Different clouds are indicators for different backbones, IoT sensors and QPUs

VLAN	Purpose
23	PC network
41	Dust sensors
109	Superconducting Vendor
284	Ion-Trap Vendor
2249	Management systems
2311	Guest
2934	Isolation of installation location
4000	Optical Isolation
4002	People Counter Cameras

TABLE I
LAB NETWORK VLANS

- The Quantum Access Nodes (QANs) generally run a supplier's software and serve as the interface between our QC infrastructure and the supplier's hardware.
- VLAN 4000 is the network that joins two QANs with the instrument rack of the SC system. The switch is used to convert the copper Ethernet into optical, for the purpose of electrical isolation of the instrument rack. In this case, there are two control systems installed in one rack, so two QANs are in this VLAN. Such a copper-to-optical bridge is also used in the case of linking one QAN to a QC station.
- Any hardware connected to the network switch in the superconducting QC instrument rack can only be accessed from the QANs in VLAN 4000. This becomes relevant when we look at the telemetry sensors.
- Our dust sensors have been procured as part of a site-wide project. These are connected to a separate VLAN that is used to gather data from those sensors to an external Prometheus instance.
- Our QC vendors each have a direct link to their devices, and use firewall devices to protect their systems.
- The Raspberry Pi used for telemetry is connected to the same network also used for the electronic scales that measure liquid nitrogen consumption.
- VLAN 4002 connects a second NIC in a Windows PC to the subnet that also serves the Axis people-counter cameras. While nearly all of our VLANs are untagged, there is a tagged VLAN 4002 connection available on one of the QAN connections - this allows the QAN node to also access the people-counter cameras directly.
- Our DCDB server sits in yet another VLAN, accessible from within our institutional backbone.

B. A QAN as telemetry data collector

As already mentioned, the Telemetry Raspberry Pi has many directly connected sensors plus some network-connected sensors and delivers that data to our DCDB instance. However, there are other sensors that cannot be reached from the RPI, and a number of those sensors are inside a vendor's rack. For example, a typical QC control rack will contain a smart PDU, and a UPS, and the active rack-cooling also can be connected to the network (MODBUS TCP). All of these devices can deliver data points that we're interested in.

To this end, a "*dcdbpusher*" service runs on a QAN. As a part of the DCDB software suite, the dcdbpusher gathers data from those devices only reachable from the QAN, and forwards it to the DCDB server.

Of course, since the QAN also runs the vendor's software, this can also be a point from which data can be read from the vendor's API and passed on to DCDB. Another example is the case of the dust sensors, where data is gathered by an external Prometheus instance, and then forwarded to the DCDB server.

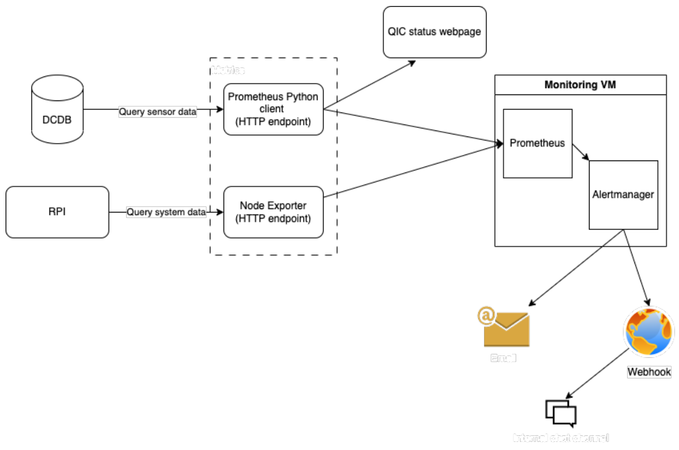


Fig. 7. Monitoring system architecture

V. Data Monitoring and Sensor Notification

In this section, we present a comprehensive approach to data monitoring and sensor notifications, as shown in Fig. 7. Notifications are generated by the system under the following conditions:

1) When data transmission interrupts (status alerts).
2) When a sensor value exceeds or falls below predefined thresholds (threshold alerts).

These notifications are facilitated by the integration of open source tools, Prometheus and Alertmanager [18], which provide robust monitoring and alerting capabilities.

A. Prometheus & Alertmanager Integration

To ensure reliable monitoring and timely notifications, our system uses Prometheus and Alertmanager for data collection, storage and alerting. Prometheus continuously scrapes metrics from various sources and evaluates predefined rules to detect the critical conditions. When such conditions are met, alerts are triggered and sent to Alertmanager, which manages alert delivery and escalation.

*a) **Metrics**:* Metrics is the way Prometheus collects data for monitoring. It provides real-time insights into the system's or database or sensors performance and health. In our system, metrics are generated through two primary mechanisms.

1) **Prometheus Python Client**: we utilize a python script leveraging Prometheus client library [18] to collect sensor data from the DCDB. This data is converted into two distinct types of numerical metrics.
   - **Status Metrics:** A Boolean metric that indicates whether data has been received within a specific time interval (1 for data received, 0 otherwise).
   - **Threshold Metrics:** A metric that records the actual value of the sensor data.
2) **Node Exporter**: To monitor the health and status of devices, such as the Raspberry pi (RPI) units deployed in the system, Node Exporter [19] is deployed. It collects system level metrics, including:
   - Device activity (active/down status).
   - Resource utilization, such as CPU, memory, and storage.

These metrics are then scraped by Prometheus and integrated into the monitoring framework.

*b) **Prometheus Workflow**:* Prometheus scrapes the metrics generated by the Python script and Node Exporter through its HTTP endpoints [6]. These metrics are stored in a time-series database and evaluated against alerting rules using PromQL, Prometheus query language [18]. Examples of rules include:

- Detecting sustained high sensor value (e.g., temperature of the cryostat is above a threshold value for 5 minutes).
- Identifying access to the lab after allowed times. (e.g., door sensor was triggered, i.e., lab was accessed after the allowed times).
- Identifying device failures (e.g., Raspberry pi not reporting metrics within a specific time frame).
- Sensor missing data (e.g., Temperature sensor does not receive any data for one hour).

*c) **Alertmanager**:* The triggered alerts are forwarded to Alertmanager, which manages the notification process. This includes:

- Configuring delivery methods (e.g., email, webhooks, or internal chat channels).
- Managing timing (e.g., suppressing alerts during maintenance windows).
- Silencing redundant alerts and escalating critical once to ensure timely response.

The machine hosting Prometheus and Alertmanager is also monitored by a Checkmk program, ensuring the overall monitoring solution is highly available and efficient. Alerts originate from actual incidents encountered during system operations: whenever a malfunction occurs or a metric exceeds its configured threshold, a corresponding alarm is generated. This alarm is then integrated into the sensor reporting workflow, facilitating proactive maintenance measures and improving the system's operational reliability.

B. Data Visualization

*a) **Status Page**:* A status page has been developed to provide an overview of the environmental sensors' state. This page displays the status of all sensors and records the timestamp of the last received value. It is built using the status metrics generated by the Prometheus client in the Python script.

*b) **Grafana**:* We are using Grafana as a primary tool for data visualization. The DCDB is connected to the Grafana and we have created dashboards to visualize the sensors data, as described in Sec. 2.

VI. Software Architecture

Data received from DCDB are stored and retrieved in an integer format as we can save significant space, especially for telemetry systems where thousands of readings are being regularly collected. However, we cannot feed this format directly into our machine learning models. Instead, we require a pipeline to automate the extraction of a clean dataset from DCDB for Machine Learning (ML) applications, specifically for real-time applications as mentioned in Sec. VII-A. The pipeline shown in Fig. 8 is designed specifically for machine learning applications targeting the optimization of different quantum computing metrics (e.g., fidelity analysis, mid-circuit measurements).

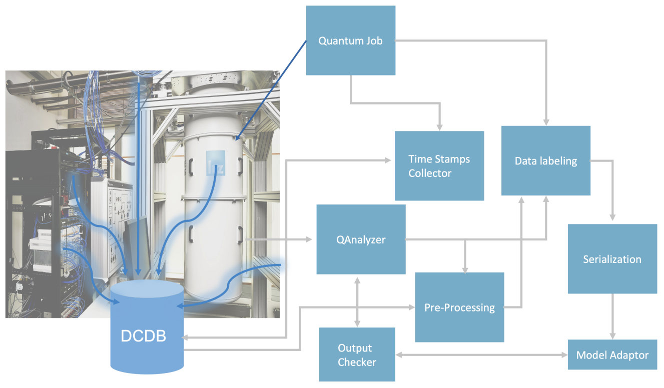


Fig. 8. Software architecture showing the pipeline for data collection and transformation from DCDB to prepare them for use in real-time analysis machine learning models

A. Software Components

- **DCDB (Data Center Database)**: The database that stores all the telemetry values in a time series manner [7].

- **Preprocessing**: The preprocessing consists of a DCDB modifier that converts integer values from the DCDB into floats with the correct identified units. This conversion is crucial for ensuring that the data is accurately represented and usable in further processing steps. The data will then be normalized and standardized to prepare it as suitable input for use in machine learning models.
- **QAnalyzer**: QAnalyzer has several quantum analysis components, like the Fidelity Analysis component, which performs a basic fidelity check by running two Hadamard gates as a cron job every half hour. This check verifies that the system is functioning correctly and provides a basic indication of system performance. The fidelity analyzer can be adapted to incorporate more advanced fidelity analysis techniques, providing a deeper understanding of system performance and identifying areas for improvement. It can also include cost functions from the output checker to enable data reduction techniques to optimize the management of the data earlier in the machine learning pipeline.
- **Time Stamps Collector**: The Timestamp Collector queries the timestamps of quantum jobs to retrieve relevant changes of events data from the DCDB. By collecting data between the start and end timestamps of each submitted quantum job, this component ensures that the analysis is based on an accurate data collection period.
- **Output Checker**: According to the model adapter, an output checker can be specified for faster execution, allowing the system to ignore outliers in results and skip to the next one when necessary. This approach is particularly crucial for some quantum mid-circuit measurement techniques to stop the process of running circuits if there is skewing from expected results.
- **Data Labeling**: This component processes the collected data, normalizing and formatting it for use in machine learning models. It ensures that the data is clean and ready for analysis. The labeled data can be output in various formats: directly to the end user, submitted to the job runner, or passed to a profiler for correlation analysis.
- **Serialization**: Data serialization formats the data for robust transfer to the machine learning model. This step ensures that the data is transmitted accurately and efficiently, minimizing the risk of errors during the transfer process.
- **Model Adapter**: The Model Adapter identifies all dependencies needed from the rest of the system (e.g., the output checker) and loads the data into the machine learning model. This component ensures that the model has access to all necessary data and resources for accurate and effective analysis.

This system facilitates the tailored use of telemetry data in quantum machine learning, enabling the execution of real-time analysis techniques.

VII. Telemetry Based Applications

The development of a carefully designed system and supporting tools has facilitated the effective utilization of environmental data. Key applications have emerged, including the transformation of environmental parameters into actionable features and their ranking based on importance scores relative to the Hellinger fidelity of quantum circuits.

A. *ML-based Error Mitigation Scheme*

Error-mitigation is a prominent technique in NISQ-era, that aims to improve the accuracy of the quantum computation. The mitigation techniques could be applied at various levels; namely as a pre-processing or post-processing step. For instance, applying optimized pulse sequences for a given quantum circuit [20] could enhance the solution accuracy and it acts as a pre-process of the quantum computation. On the other hand, there are also methods where there are multiple versions of the quantum circuit are prepared by inserting different numbers of gates, aiming to extrapolate to a noiseless result as a post-process technique [21], [22]. There also have been methods where the optimized gate implementation is done via learning-based error mitigation methods [23].

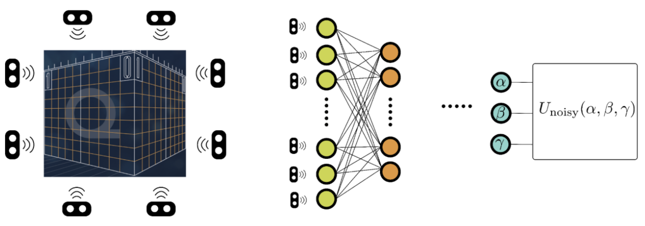


Fig. 9. The environmental values captured are continuously fed to the already trained neural network, to produce single-qubit gate mitigation unitaries, that can be fused with the single-qubit gates in the original input circuit.

In the context of telemetry, we are investigating how the environmental data could alleviate the noise on superconducting circuits and build a hybrid quantum-classical machine learning pipeline to mitigate coherent single-qubit errors. The hybrid model takes 9 different environmental parameters and produces a single qubit unitary that can then be appended to the gate to be improved, which is demonstrated in Fig. 9. This pipeline requires a seamless integration with the telemetry database, since the goal is to build the pipeline as a real-time error mitigation scheme within the quantum compilation stack. As the method uses the telemetry data as an input, and produces error-mitigated circuits on-the-fly, it works as a pre-processing step and is included in the compilation stack, which acts as the final pass after the circuit is already tailored for the target hardware.

B. *Environmental Features Ranking*

The deployment of 508 environmental sensors poses a significant scalability challenge, as replicating this dense sensor network across multiple quantum computing installations is impractical. To address this, we employ a Random Forest machine learning algorithm to identify and rank the environmental features most strongly correlated with quantum device performance. This approach enables targeted sensor deployment by prioritizing variables critical to the specific qubit technology platform. To analyze further correlation a single qubit gate experiment was implemented with two Hadamard gates on different random qubits. This experiment is implemented to investigate the effect of decoherence and dephasing errors of environmental variances, and Hellinger fidelity of GHZ state for 5 qubits mapped to the QPU architecture is defined as the target variable. Environmental features are ranked by their relative contribution to reducing variance in the GHZ state fidelity metric. This analysis reveals which parameters most strongly influence 1) single-qubit decoherence/dephasing and 2) correlated errors in two-qubit gates, providing a pathway to optimize sensor configurations for noise mitigation.

The highest importance scores are:- Cryostat Vaccum can pressure: 0.495, Single qubit gate circuit: 0.335, Room Temperature: 0.095, Mixing Champer Temperature:0.075. Identifying the most important environmental factors provides valuable insights into managing the quantum environment and exploring underlying relationships to quantum computation. The data was collected at 5-minute intervals over a 64-hour period. Seven sensors were queried at these timestamps, and the data was converted into features for use in a random forest algorithm. The total querying time for this duration was 5.365 seconds, which is faster than the 6-second update interval of the quickest sensor in the group. This shows that real-time telemetry processing is feasible given the variance of environmental parameters.

VIII. Conclusions

The integration of an IoT-based telemetry system into the quantum computer environment has been demonstrated as a scalable and efficient solution to monitor and characterize QPUs. By making use of widely available sensors and microcontrollers, we ensure efficient data collection which can be further used for alerts or for creating machine learning models to characterize and mitigate the environmental noise. Our findings highlight the importance of environmental parameters in the operation of QPUs, across all modalities, as evidenced by our deployment in a joint lab with superconduction and trapped ion systems alongside an HPC system. This shows the need to include these data as metadata for QPUs and announced fidelity results. By characterizing and integrating environmental parameters into our system, we identified specific needs for specific modalities and with that we can improve the stability and performance of the QPUs. This drives the development of an infrastructure that leads to environment-aware quantum computing software tailored specifically for co-locating quantum computers with HPC servers.

Acknowledgments

The work was funded by the Munich Quantum Valley (MQV), which is supported by the Bavarian State Government with funds from the Hightech Agenda Bayern.

References

[1] M. N. Farooqi and M. Ruefenacht, "Exploring hybrid classical-quantum compute systems through simulation," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02, 2023, pp. 127–133.

[2] M. Ruefenacht, B. G. Taketani, P. Lähteenmäki, V. Bergholm, D. Kranzlmüller, L. Schulz, , and M. Schulz, "Bringing quantum acceleration to supercomputers," May 2022, accessed: 27-03-2025. [Online]. Available: https://www.quantum.lrz.de/fileadmin/QIC/Downloads/IQM_HPC-QC-Integration-Whitepaper.pdf

[3] J. R. Cruise, N. I. Gillespie, and B. Reid, "Practical quantum computing: The value of local computation," 2020. [Online]. Available: https://arxiv.org/abs/2009.08513

[4] T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowrishankar, A. Frisch, and T. Monz, "Quantum computers for high-performance computing," *IEEE Micro*, vol. 41, no. 5, pp. 15–23, 2021.

[5] J. K. Iverson and J. Preskill, "Coherence in logical quantum channels," *New Journal of Physics*, vol. 22, no. 7, p. 073066, 2020.

[6] H. Ahmed, X. Deng, H. Heller, C. Guillen, A. Zulfiqar, M. Ruefnacht, A. Jamadagni, M. Tovey, M. Schulz, and L. Schulz, "Quantum computer metrics and hpc center environmental sensor data analysis towards fidelity prediction," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02, 2023, pp. 154–160.

[7] A. Netti, M. Müller, A. Auweter, C. Guillen, M. Ott, D. Tafani, and M. Schulz, "From facility to application sensor data: modular, continuous and holistic monitoring with dcdb," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. ACM, Nov. 2019, p. 1–27. [Online]. Available: http://dx.doi.org/10.1145/3295500.3356191

[8] "Prometheus node exporter on raspberry pi: How to install," accessed: December 5, 2024. [Online]. Available: https://linuxhit.com/prometheus-node-exporter-on-raspberry-pi-how-to-install/

[9] "Sensorb h17 loudness sensor," accessed: December 5, 2024. [Online]. Available: https://cdn-reichelt.de/documents/datenblatt/A300/SENSORB%20H17

[10] "Loudness sensor, grove," accessed: December 5, 2024. [Online]. Available: https://wiki.seeedstudio.com/Grove-LoudnessSensor/

[11] "Grove pi plus," accessed: December 5, 2024. [Online]. Available: https://wiki.seeedstudio.com/GrovePiPlus/

[12] "16-bit 3-axis magnetometer, 1.7-1.9v smd bmm350," accessed: December 5, 2024. [Online]. Available: https://cdn-reichelt.de/documents/datenblatt/C900/BST_BMM350_FL000-00.pdf

[13] "Datasheet ds3143," accessed: December 5, 2024. [Online]. Available: https://bartingtondownloads.com/wp-content/uploads/DS3143.pdf

[14] "Datasheet ds2519," accessed: December 5, 2024. [Online]. Available: https://www.bartingtondownloads.com/wp-content/uploads/DS2519.pdf

[15] "High-precision ad hat," accessed: December 5, 2024. [Online]. Available: https://www.waveshare.com/wiki/High-Precision_AD_HAT

[16] "Ads1263 - precision adc by texas instruments," accessed: December 5, 2024. [Online]. Available: https://www.ti.com/product/ADS1263

[17] "Mwn-netzkonzept 2019," accessed: December 5, 2024. [Online]. Available: https://www.lrz.de/services/netz/mwn-netzkonzept/mwn-netzkonzept-2019.pdf

[18] "Prometheus Python Client," github.com/prometheus/client_python, 2015, accessed: December 05,2024.

[19] "Node Exporter," github.com/prometheus/node_exporter, 2014, accessed: December 5,2024.

[20] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Physical Review Letters*, vol. 82, no. 12, p. 2417, 1999.

[21] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Physical Review X*, vol. 7, no. 2, p. 021050, 2017.

[22] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Physical review letters*, vol. 119, no. 18, p. 180509, 2017.

[23] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, "Error mitigation with clifford quantum-circuit data," *Quantum*, vol. 5, p. 592, 2021.