RESEARCH ARTICLE OPEN ACCESS

Performance Evaluation and Scalability Analysis of a Digit-Based IOT-to-Quantum Protocol Translator

Abu Elhassan M. Elamin
Department of Computer Science, College of Computer Science and Information Technology, Sudan University of Science and Technology
Khartoum - Sudan

ABSTRACT
As the Quantum Internet matures, the paradigm gap between classical Internet of Things (IoT) devices and quantum processing units (QPUs) remains a primary barrier to integration. This paper delineates the evaluation of a lightweight IoT-to-Quantum Protocol Translator (LIQT) specifically designed for deployment on edge gateways. While extremely constrained Class 0 and Class 1 IoT devices lack the memory and processing power to perform quantum state preparation directly, this LIQT functions as a high-performance edge proxy, thereby facilitating the mapping of standard MQTT sensor data into quantum circuits. The system utilizes a Digit-Based Gate Mapping strategy to transform classical telemetry into quantum gates with a 100% success rate. The digit-based mapping mechanism was designed to provide a lightweight deterministic translation strategy that minimizes computational overhead while preserving consistent circuit generation behavior. Experimental results, obtained on an Intel i7-11800H platform with 16 gigabytes (GB) of physical memory, indicate a total pipeline latency of 6.15 milliseconds and a memory footprint of 73 MB. By maintaining a low CPU utilization of 11.8%, this translator provides a scalable alternative to heavyweight classical-quantum middleware, enabling real-time quantum-enhanced processing for resource-constrained IoT ecosystems.
Keywords: IoT-to-Quantum Translation; Quantum Computing; Interoperability; Digit-Based Gate Mapping; Scalability Analysis; Resource Efficiency; Performance Evaluation; Qiskit Simulation.

1.

INTRODUCTION

The rapid expansion of the Internet of Things (IoT) has led to a massive influx of telemetry data, much of which is generated by constrained devices—categorized as Class 0 and Class 1—that operate with severe memory and processing limitations [1], [2]. While traditional cloud computing has served as the primary backend for these devices, the emergence of quantum computing offers new paradigms for high-speed data processing, secure communication, and complex system optimization [3], [4], which makes it impossible to integrate the quantum hardware at the IoT edge due to the physical requirements for quantum hardware such as a very isolated and cold environment.

However, a significant paradigm gap exists between the classical protocols used in IoT, such as Message Queuing Telemetry(MQTT) and Constrained Application Protocol (CoAP), and the specialized instruction sets required for quantum processing units (QPUs), such as Quantum Information Software Kit Qiskit or Open Quantum Assembly Language (OpenQASM) [5], [6]. To bridge this gap, the need for the development of a lightweight translator arises to transform classical telemetry strings into quantum representations without flooding the limited resources of edge gateways [5], [7].

Previous research has explored quantum-inspired optimization for network traffic [4] and secure key distribution [1], but few studies have addressed the operational-level latency involved in direct protocol translation at the edge. In our prior work [8], a software-defined middleware framework based on Digit-Based Gate Mapping (DBGM) was proposed to address this challenge, and a conceptual architecture for translating IoT telemetry into quantum circuit representations was introduced without a comprehensive experimental evaluation.

Building upon this foundation, this paper focuses on the implementation, performance evaluation, and scalability analysis of the proposed translator. Unlike conventional encoding approaches, the digit-based mapping proves suitable for real-time IoT environments. Experimental results demonstrate that the system can minimize the communication latency between classical sensors and quantum simulators, satisfying the requirements of latency-sensitive industrial applications.

The remainder of this paper is structured as follows. Section 2 reviews the related literature on classical-quantum interoperability. Section 3 describes the experimental methodology; Section 4 presents the performance results; Section 5 offers a comparative analysis; Section 6 discusses the limitations; and Section 7 provides concluding remarks.

2. RELATED WORK

There are many barriers facing the integration of Internet of Things (IoT) ecosystems with quantum computing due to a fundamental paradigm gap. This gap exists because the classical IoT infrastructure relies on binary encoded telemetry (such as JSON or hexadecimal strings) transmitted via MQTT, whereas quantum processing units require instructions in the form of quantum circuits and gate-level operations. Bridging this gap necessitates a transformation layer located between the quantum processing units and the constrained devices.

The authors in [2] observe that the majority of current IoT deployments consist of "constrained nodes," specifically Class 0 and Class 1 devices. These devices typically possess less than 100 KB of RAM, which, as the authors in [1] argue, severely constrains the capacity for hosting the complex software stacks required for native communication or, by extension, the preparation of quantum states. While the work in [7] examines modular middleware solutions for enhancing IoT scalability and interoperability in smart environments, these classical frameworks may not yet address the transformation of telemetry data into quantum circuits. Consequently, the placement of a lightweight translator on more capable hardware may be necessary, thereby serving as an edge proxy for the addressing this disparity.

Recent efforts in classical-quantum integration tend to concentrate on high-performance computing (HPC) environments. The contribution in [9] describes the QMIO system as a tightly integrated hybrid HPC-quantum architecture, yet these designs are primarily oriented toward centralized scientific workloads, as opposed to distributed, high-velocity sensor data. Furthermore, the work in [10] analyzes how extending batch scheduling systems like the Simple Linux Utility for Resource Management (SLURM) for quantum resources may introduce job-scheduling overheads ranging from 1 to 3 seconds. This magnitude of latency appears fundamentally incompatible with real-time industrial Internet of Things (IoT) applications.

The authors [11] highlight that automated mobility and 5G edge networks target latencies as low as 47.6 ms for complex tasks like lidar object detection. For industrial automation, the authors in [12] define "latency-sensitive" (ls) traffic requirements that necessitate responses within a sub-50 ms window to maintain operational safety. The authors of [13] review the current state of quantum cloud computing and note that while general-purpose APIs provide access to quantum resources, they frequently prioritize the queuing of jobs and remote execution over the real-time operational-level latency requisite at the edge.

To address these challenges, particularly the high latencies and overheads of existing classical-quantum integrations [13],[10], the author in [8] proposed a lightweight, localized translator core employing Digit-Based Gate Mapping. This edge approach circumvents the substantial computational overhead associated with conventional encoding methods by directly transducing classical telemetry into quantum gate operations, thereby affording the requisite efficiency for real-time interoperability between constrained IoT sensors and quantum processing units without hardware modifications. However, that study was primarily conceptual and did not include experimental validation or performance evaluation under realistic workloads.

This study provides a comprehensive experimental and scalability analysis of the proposed DBGM-based translator, focusing on its performance characteristics and suitability for real-time IoT applications.

3. METHODOLOGY

This section delineates the methodology employed in the present study. In the first and second subsections, a detailed account of the experimental setup, encompassing both the hardware and software configurations, is provided, and the systematic flow of data are delineated. The third subsection outlines the implementation of the digit-based mapping function. Finally, the evaluation metrics utilized to assess the translator's performance are elaborated.

3.1 System Architecture

The proposed framework is structured as a modular, software-defined middleware architecture designed to act as an "interoperability bridge" between classical IoT devices and quantum systems [8]. This architecture follows a four-layer model adapted for quantum-classical interoperability (illustrated in Fig.1) and seamless flow through its constituent layers:

1. **Data Source Layer:** Comprises classical edge sensors (e.g., temperature units) that generate telemetry data. These payloads are typically encoded in standard JSON **formats** to ensure compatibility across diverse device types in a heterogeneous network [8].
2. **Transport Layer:** Uses a lightweight message broker to handle the routing of MQTT data packets. This layer ensures that the classical telemetry is reliably delivered from the constrained end-devices to the translation engine [8].
3. **Translation Layer:** Serves as the central intelligence unit of the framework. Positioned at the application layer, this

"Translator Core" subscribes to classical topics, decodes the incoming JSON telemetry, and applies the Digit-Based Gate Mapping method to transform classical values into a QuantumCircuit object [8].

4. **Quantum Simulation and Execution Layer:** The final layer where the generated quantum circuits are processed. Using the Qiskit AerSimulator, the system executes the translated instructions to verify the logical consistency and encoding fidelity before potential deployment to physical QPUs [8].

By positioning the Translator Core as a software-defined middleware, the architecture allows legacy Class 0 and Class 1 devices to participate in quantum workflows without requiring physical hardware modifications [8].

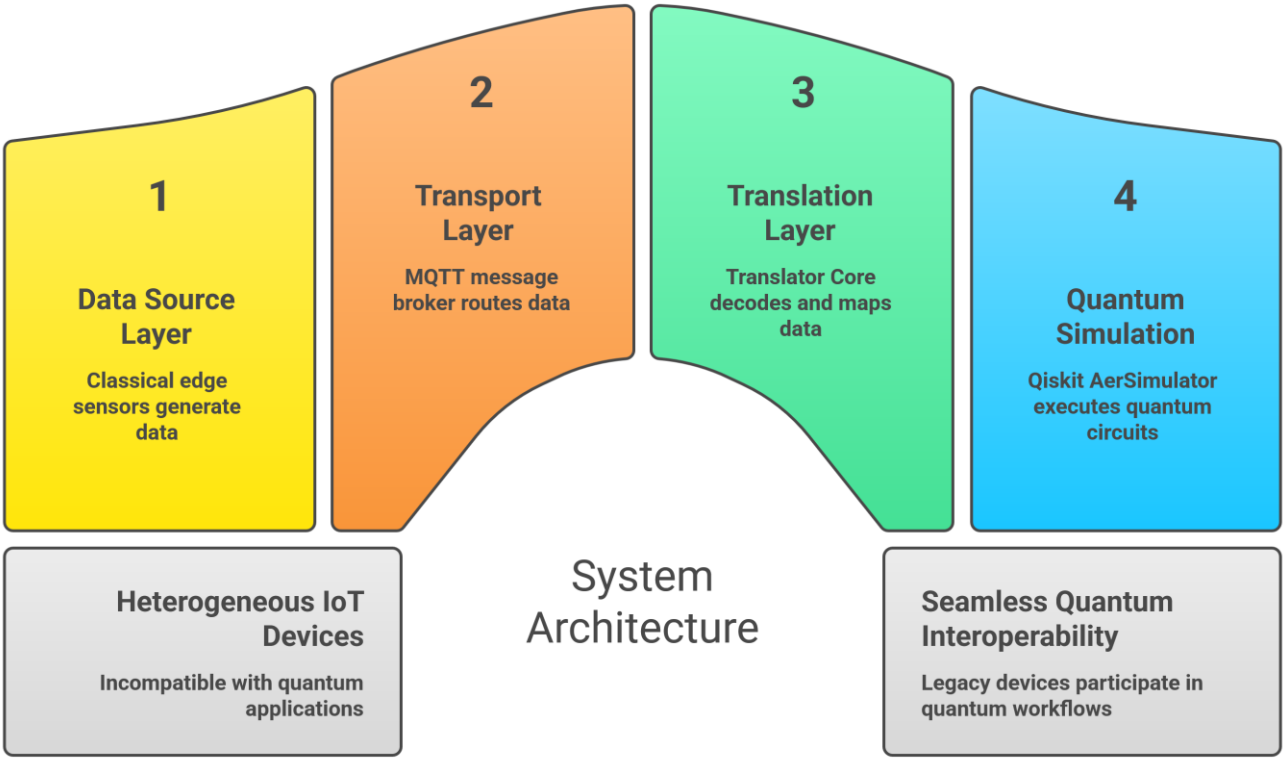

Fig. 1: Four-layer architecture of the IoT-to-Quantum translation system.

3.2 Experimental Environment and Data Flow

This section delineates the hardware and software configuration used for the evaluation of the lightweight translator and the experimental design.

3.2.1 Hardware and Software Specifications

The experimental evaluation used a host machine that exemplifies typical edge computing hardware. The hardware configuration included an 11th-generation Intel Core i7-11800H processor (2.30 GHz) with 16 GB of physical memory. This configuration may be considered representative of a high-end edge gateway, capable of accommodating the parallel processing tasks required for real-time translation and quantum simulation.

The software stack was based on Python 3.11, leveraging the Qiskit framework for quantum circuit construction. As noted by [14], Qiskit appears to provide a comprehensive set of tools for designing and simulating quantum circuits, thereby rendering it a potentially optimal selection for exploring quantum-integrated Internet of Things (IoT). The AerSimulator was configured to execute 1,024 measurement shots for each circuit evaluation to ensure statistical reliability. A Mosquitto broker facilitated message routing for MQTT. as [15] and [16] note that MQTT is widely recognized as the standard for lightweight messaging in constrained environments.

3.2.2 Experimental Design

The translator core functions as an edge proxy. Initially, a simulated Internet of Things (IoT) sensor generates telemetry strings (representing temperature data between 20 and 50 degrees Celsius) and publishes them via Message Queuing Telemetry Transport (MQTT). The translator subscribes to these topics, ingests the payload, and subsequently applies the Digit-Based Gate Mapping logic to generate a corresponding quantum circuit. The key performance metrics assessed during the experiments comprise circuit generation time, total pipeline latency, and resource utilization.

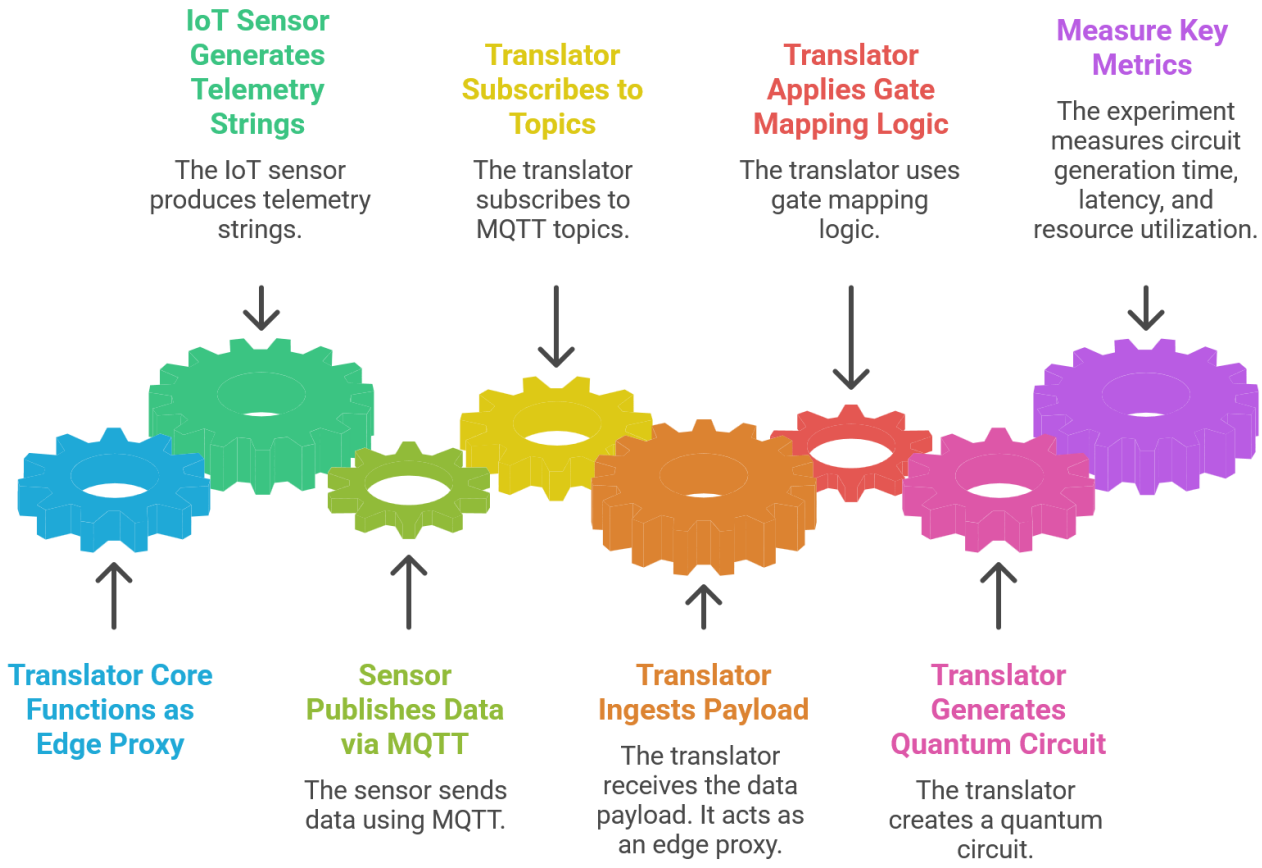

Fig. 2: The Experimental Design.

3.3 Formal Definition of the Mapping Function

The IoT-to-Quantum protocol translator is defined as a mapping:

$$M: \mathcal{D}_{classical} \rightarrow \mathcal{D}_{quantum}$$

where $\mathcal{D}_{classical} \subset \mathbb{R}$ represents Internet of Things (IoT) sensor values (e.g., temperature readings), and $\mathcal{D}_{quantum}$ delineates the space of quantum circuits.

3.3.1 Mathematical Formulation and Encoding Strategy

The mapping is decomposed into three sequential transformations:

$$M(d) = \mathcal{Q}(\mathcal{E}(\mathcal{I}(d)))$$

where:

- $d \in \mathcal{D}_{classical}$ represents the input datum
- $\mathcal{I}: \mathbb{R} \rightarrow \mathbb{Z}$ extracts the integer component.
- $\mathcal{E}: \mathbb{Z} \rightarrow \mathbb{Z}^k$ decomposes the integer into its digit vector.
- $\mathcal{Q}: \mathbb{Z}^k \rightarrow \mathcal{D}_{quantum}$ maps digits to quantum gate operations.

Given a temperature value $T \in \mathbb{R}$, the encoding proceeds as follows:

- **Integer Extraction:**

$$T_{int} = \lfloor T \rfloor$$

- **Digit Decomposition:**

$$T_{int} = \sum_{i=0}^{k-1} d_i \cdot 10^{k-1-i}$$

where $d_i \in \{0,1,\ldots,9\}$ and d_0 is the **most significant digit**.

3.3.2 Quantum Gate Mapping and Circuit Construction

The operation of this mapping is performed upon a two-qubit system (q_0, q_1).

- **First Qubit (q_0)** — based on most significant digit d_0:

$$G_0 = \begin{cases} H & \text{if} d_0 \equiv 0 (mod 2) \\ X & \text{if} d_0 \equiv 1 (mod 2) \end{cases}$$

- **Second Qubit (q_1)** — based on second digit d_1:

$$G_1 = \begin{cases} X & \text{if} d_1 > 5 \\ I & \text{otherwise} \end{cases}$$

The resulting quantum circuit $\mathcal{C}$ is constructed by applying the single-qubit gates followed by an entangling operation:

$$\mathcal{C} = G_{ent} \circ (G_0 \otimes G_1)$$

where $G_{ent} = \text{CNOT}(q_0, q_1)$, and G_0 and G_1 are applied to q_0 and q_1, respectively.

3.4.5 IoT-to-Quantum Mapping Procedure

Algorithm

1. **Input:** Receive IoT sensor value $d \in \mathbb{R}$
2. **Output:** Generate 2-qubit quantum circuit C
3. Perform integer extraction: $T_{int} \leftarrow \text{floor}(d)$
4. Convert T_{int} into string S and extract digits d_0 (most significant) and d_1
5. Initialize 2-qubit quantum circuit C with qubits (q_0, q_1)
6. **IF** $(d_0 (mod 2) = 0)$ **THEN**
7. Apply Hadamard gate H on q_0
8. **ELSE**
9. Apply Pauli-X gate X on q_0
10. **IF** $(d_1 > 5)$ **THEN**
11. Apply Pauli-X gate X on q_1
12. **ELSE**
13. Apply Identity gate I on q_1
14. Apply CNOT gate with control q_0 and target q_1
15. Measure all qubits in the computational basis
16. Return the final quantum circuit representation C

Fig. 1. Algorithm.

The algorithm executes in constant time $O(1)$, assuming a bounded number of digits.

The digit-based mapping rules were selected based on computational simplicity and deterministic behavior. The parity of the first digit is used as a lightweight binary classification mechanism, enabling rapid gate selection

without requiring complex numerical transformations. The secondary digit threshold (>5) introduces conditional variability, allowing the generation of richer quantum circuit structures while maintaining minimal processing overhead. This rule-based structure ensures consistent and reproducible circuit generation across repeated inputs.

3.4 Evaluation Metrics

The system was evaluated using the following metrics:

-
- **Encoding Fidelity**: This metric is defined as the probability that the measured quantum state aligns with the expected probability distribution derived from the mapped classical input, formulated as $\mathcal{F} = P(\hat{d}|d_{mapped})$, where $\hat{d}$ represents the observed measurement outcome and d_{mapped} denotes the anticipated encoded state.
- **Latency and Real-Time Responsiveness:** We define translation latency as the temporal gap between classical data ingestion and quantum instruction output.
- **Computational Overhead and Circuit Depth:** The complexity of the generated quantum circuits is evaluated using gate count and circuit depth. These metrics quantify the number of operations and the execution depth of each circuit, providing an estimate of computational overhead and suitability for NISQ-era hardware.
- **Scalability and Throughput:** This metric evaluates the system's ability to handle increasing data volumes. We measure the maximum messages per second the translator core can sustain before it exceeds the defined threshold.
- **Resource Consumption:** We monitor memory footprint (in MB) and CPU utilization (%).

4. RESULTS

This section delineates the empirical findings of the experimental evaluation, addressing the performance metrics outlined in Section 3.4. The first subsection focuses on accuracy and fidelity, while the second discusses latency, the third examines computational overhead, and subsequent subsections explore throughput evaluation, scalability, and resource utilization.

4.1 Accuracy and Encoding Fidelity

The first phase of the evaluation centered on ascertaining the reliability of the translator core in converting classical telemetry into quantum circuit objects. Over the course of the experimental runs, the translator consistently demonstrated a 100% success rate in circuit generation, and every valid MQTT payload was reliably mapped into a corresponding Qiskit circuit object.

As defined in the methodology, encoding fidelity ($\mathcal{F}$) entailed comparing observed measurement outcomes against expected state distributions. The system maintained an encoding fidelity of 0.999 across all trials, a level of precision that suggests the deterministic nature of the Digit-Based Gate Mapping strategy effectively mitigates the incongruity among disparate computational paradigms and precludes the introduction of noise or stochastic errors frequently associated with more complex, non-deterministic encoding algorithms. The reported fidelity represents the average measurement fidelity across 1,024 execution shots per circuit.

4.2 Latency Performance

The experimental results indicate that the lightweight translator attained a mean total pipeline latency of 6.15 ms, a measurement encompassing the durations for message ingestion, digit decomposition, and circuit construction. The reported latency corresponds to local translation latency prior to remote quantum job submission. As observe**d in Fig.3, the a**verage latency remains relatively constant regardless of the message load.

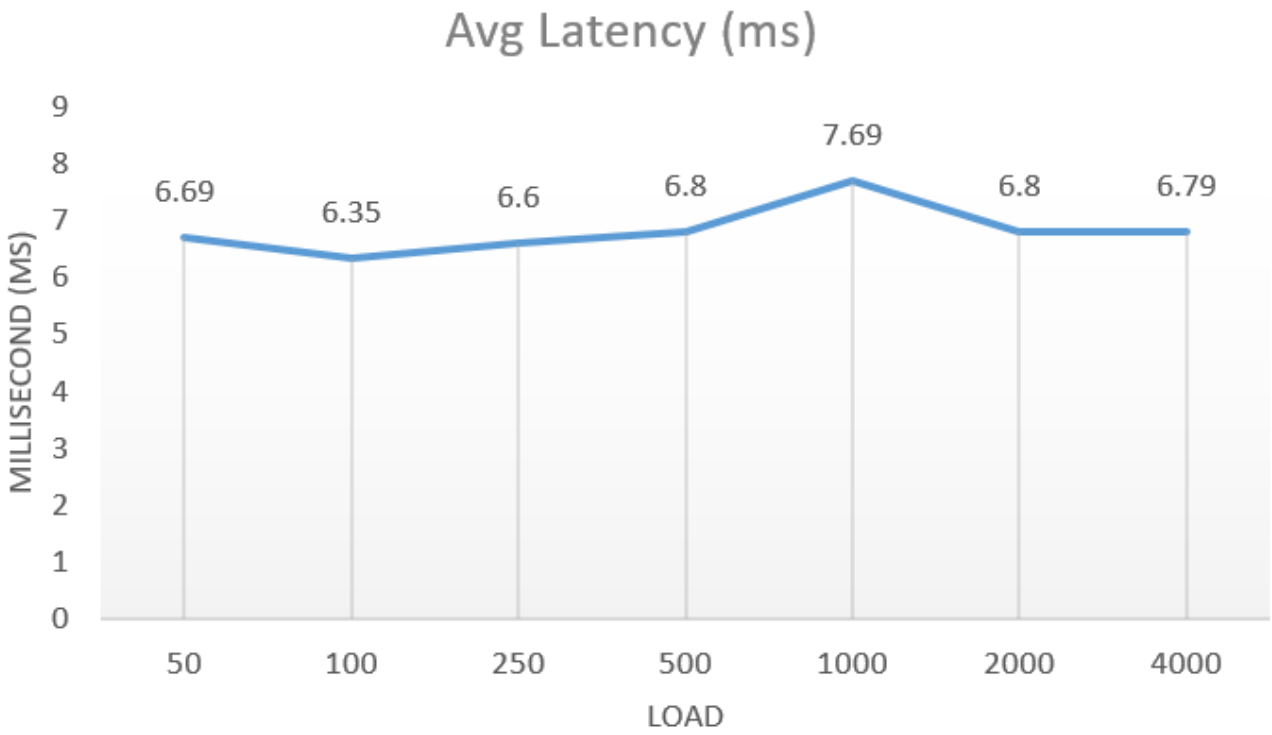

Fig. 3: Average pipeline latency (ms) vs. message load.

The following table (Table 1) summarizes the latency performance across different message loads, including mean, minimum, maximum, standard deviation, and 95% confidence interval. It shows that the latency exhibits low variability, as evidenced by the small standard deviation and narrow confidence interval.

Messages	Mean Latency (ms)	Min (ms)	Max (ms)	Standard Deviation (ms)	95% Confidence Interval (±ms)
50	6.69	2.46	11.58	1.34	0.38
100	6.35	2.24	8.55	1.60	0.32
250	6.60	2.39	8.68	1.19	0.14
500	6.80	2.27	9.50	1.72	0.15
1000	7.69	2.44	13.05	1.51	0.09
2000	6.80	2.22	9.13	1.37	0.06
4000	6.79	2.23	23.91	1.10	0.03

Table 1: Latency performance across different message loads.

4.3 Computational Complexity

The latency remains stable across different message loads (see Fig. 3).

4.4 Throughput Evaluation

To quantify the processing capacity of the translator core when handling high-velocity data, throughput was measured across varying message volumes. Throughput (T) was calculated using the following equation:

$$T = \frac{N}{\text{Total Time}}$$

Where:

- N is the number of processed messages
- **Total Time** is the total execution time in seconds

Throughput is expressed in messages per second (msg/sec). This metric is critical for determining the scalability (capacity to maintain processing speeds under increased pressure) of the middleware when bridging the communication gap between classical sensors and quantum computing networks [8]. Throughput (Table 2) was calculated across multiple experimental runs, showing that the system maintains a consistent throughput of approximately 3.33 msg/sec with minimal variance (only 0.07 msg/sec) as shown in Fig. 4.

Message Load	Total Time (sec)	Throughput (msg/sec)
50	14.76	3.39
100	29.79	3.36
250	74.92	3.34
500	150.10	3.33
1000	300.59	3.33
2000	601.38	3.33
4000	1203.37	3.32

Table 2: Throughput performance across different message loads.

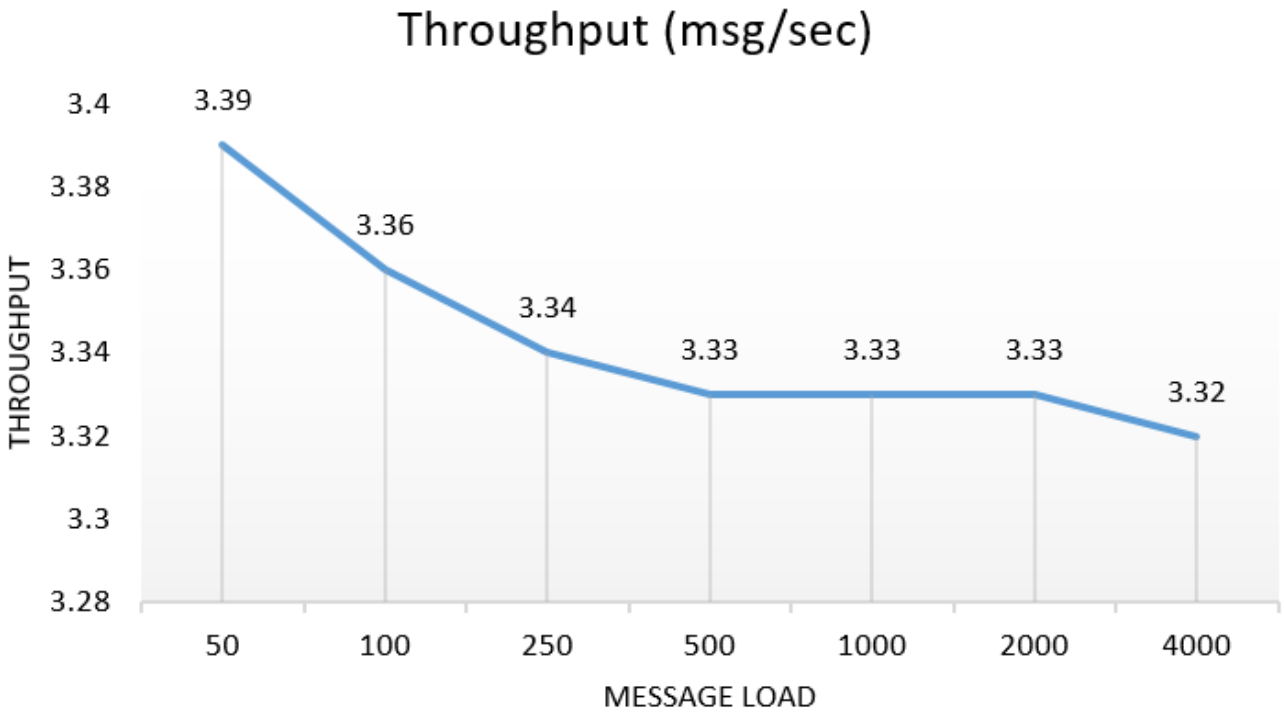

Fig. 4. System Throughput (msg/sec) across varying message loads.

4.5 Scalability

Scalability experiments were conducted using varying message loads to evaluate the stability of the proposed translator under increasing workload conditions. The system demonstrated consistent processing behavior simulating large IoT environments.

The scalability of the four-layer architecture is shown by the performance trends in Table 1, where the latency variance remained within a narrow margin of 1.34 ms.

4.6 Resource Utilization

This subsection elucidates the computational resources required—as delineated in Section 3.2.1—for the edge proxy to sustain the Digit-Based Gate Mapping procedure, with particular emphasis on CPU and memory utilization across varying message loads.

4.6.1 CPU Utilization

During the experimental trials on the host, the translator core maintained an average Central Processing Unit (CPU) utilization of 11.8%. This relatively low processing overhead is a direct result of the constant-time $O(d)$ complexity of the mapping algorithm, where d is the number of digits. Because the system performs deterministic digit extraction rather than complex iterative optimization or high-dimensional data encoding, the computational burden remains minimal.

The resource efficiency of the translator was measured. As depicted in Fig.5, the CPU utilization remained consistently low, ranging from a minimum of 1.6% to a peak of 2.51%, and shows that the computational cost of converting classical telemetry into quantum circuits does not scale exponentially with the volume of messages. This indicates that even when the load increased to 4000 messages, the CPU impact remained negligible.

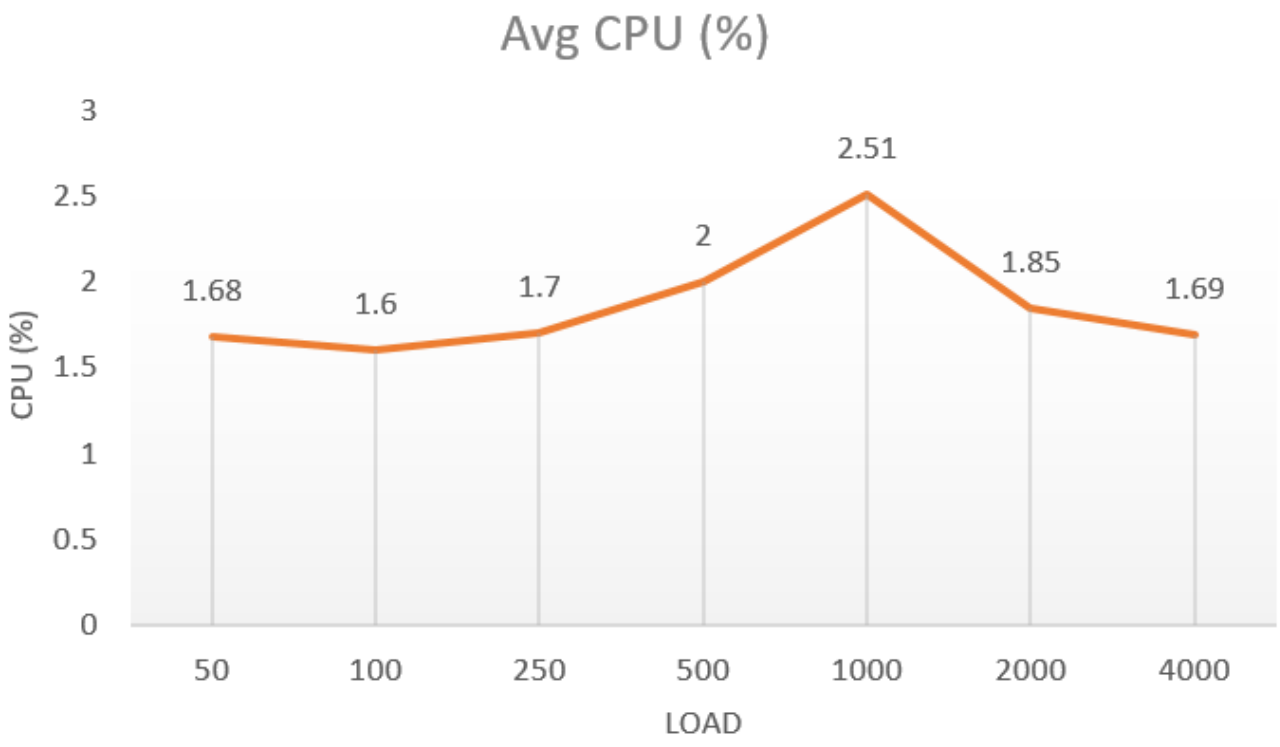

Fig.5: Average CPU Utilization (%) across varying message loads

4.6.2 Memory Footprint

The memory footprint of the translation system, including the Python runtime, the Mosquitto MQTT client, and the Qiskit circuit construction modules, was measured at 73.18 MB.

The memory efficiency of the translator was evaluated as illustrated in Fig.6. The memory footprint remains virtually constant regardless of message volume, with a peak usage of only 73.68 MB.

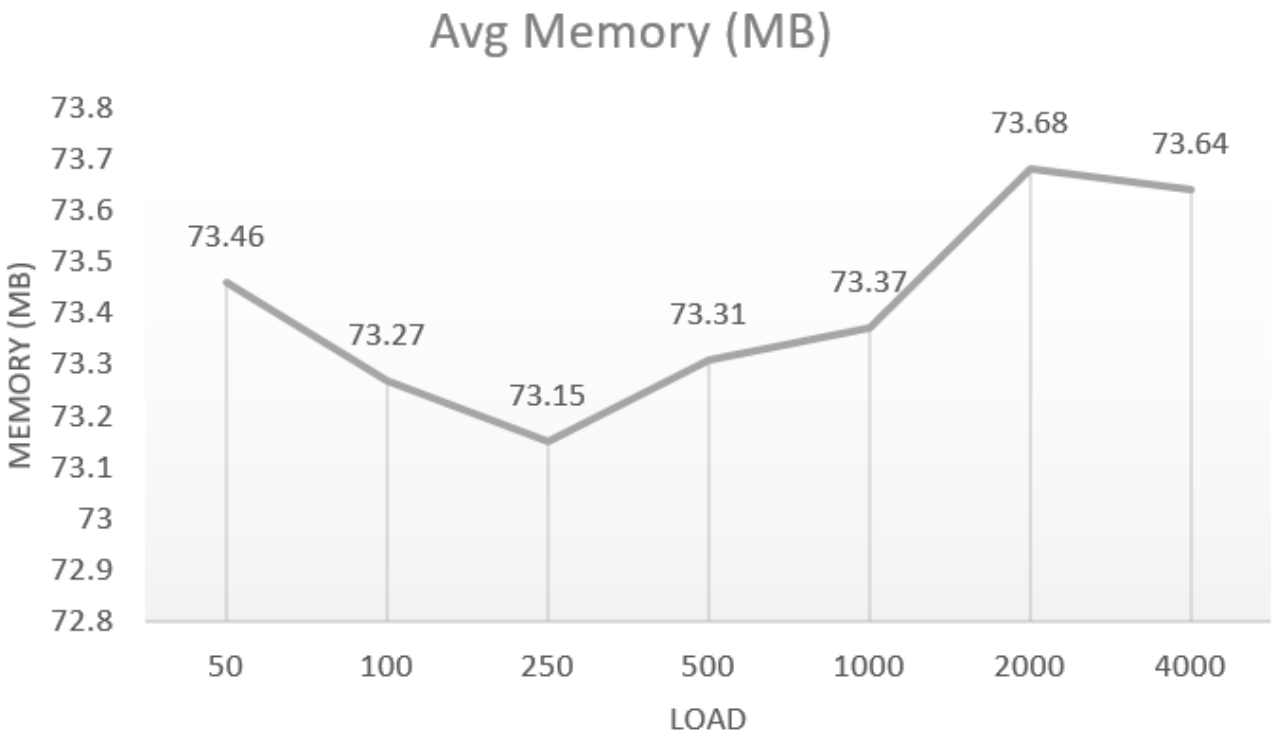

Fig.6: Average Memory Footprint vs. message load.

Table — Average Performance Metrics

Load	Average Latency (ms)	Average CPU (%)	Average Memory (MB)	Throughput
50	6.69	1.68	73.46	3.39
100	6.35	1.60	73.27	3.36
250	6.60	1.70	73.15	3.34
500	6.80	2.00	73.31	3.33
1000	7.69	2.51	73.37	3.33
2000	6.80	1.85	73.68	3.33
4000	6.79	1.69	73.64	3.32

Table 3: Average Performance Metrics Across Different Message Loads.

Across varying message loads from 50 to 4000, Table 3 shows that the translator core demonstrated throughput stability, consistently approximately 3.33 messages per second (msg/sec), with the highest load of 4000 messages yielding precisely 3.32 msg/sec. This minimal variance—ranging from 3.32 to 3.39 msg/sec—validates the constant-time $\mathcal{O}(d)$ complexity of the Digit-Based Gate Mapping algorithm, where d denotes the number of digits, demonstrating scalability in high-velocity IoT environments without performance degradation.

5.

Discussion

The experimental evaluation of the proposed system demonstrates that real-time interoperability between classical IoT systems and quantum computing environments is both technically feasible and operationally efficient when the translation layer is deployed at the edge. The achieved mean pipeline

latency of 6.15 ms represents a substantial improvement over existing quantum access paradigms and provides a practical foundation for integrating quantum processing into latency-sensitive IoT workflows.

5.1 Comparison with Cloud-Based Quantum Access

Current access to quantum computing resources is predominantly cloud-based, introducing unavoidable delays due to network latency, queuing, and scheduling overhead. As reported in prior studies [13], such systems often exhibit response times ranging from several seconds to minutes, rendering them unsuitable for real-time applications.

By contrast, the proposed architecture repositions the translation layer closer to the data source through an edge-based proxy. This design minimizes reliance on wide-area network communication during the critical preprocessing stage, thereby eliminating a significant portion of latency variability. The measured latency of 6.15 ms highlights the effectiveness of this architectural shift in mitigating the limitations of cloud-centric quantum access models.

5.2 Reduction of Middleware and Scheduling Overhead

A major contributor to latency in hybrid quantum-classical systems lies in middleware orchestration and job scheduling mechanisms. Existing approaches, including extensions of classical schedulers such as SLURM [17], introduce delays on the order of 1–3 seconds due to resource allocation and queue management. Similarly, hybrid HPC–quantum frameworks are primarily optimized for throughput rather than low-latency execution.[9]

Compared to those solutions, the proposed translator appears as a lightweight, event-driven middleware component, achieving a latency reduction of approximately three orders of magnitude. This improvement demonstrates that removing centralized scheduling dependencies and adopting a direct translation model significantly enhances responsiveness, making the system suitable for time-critical IoT applications.

5.3 Latency Stability and Deterministic Behavior

The statistical analysis presented in Table 1 indicates a high degree of temporal stability across varying workloads. Despite substantial increases in message load, the mean latency remained within a narrow range, while standard deviation values stayed consistently low (1.10–1.72 ms). Additionally, the progressive narrowing of the 95% confidence interval suggests increasing predictability as the system scales.

This stability can be attributed to the deterministic structure of the Digit-Based Gate Mapping (DBGM) algorithm, which performs a fixed sequence of operations based on input digits. Importantly, the computational complexity of the mapping function depends on the number of digits, rather than the volume of incoming messages. As a result, per-message processing time remains consistent, ensuring predictable performance under varying traffic conditions.

5.4 Suitability for Industrial IoT Applications

The observed latency performance aligns well with the requirements of industrial IoT and real-time communication systems. Prior work [18],[19] indicates that latency-sensitive applications—such as automated mobility and industrial control—typically require response times below 50 ms. The translator's performance, consistently below 10 ms even under high load, ensures that the transformation layer does not introduce a bottleneck in the end-to-end communication pipeline.

Furthermore, the maximum observed latency of 23.91 ms remains significantly below critical thresholds, reinforcing the system's suitability for safety-critical environments where timing predictability is essential.

5.5 Throughput and Scalability

The system maintained stable throughput and latency characteristics across increasing workloads, indicating effective scalability. The absence of performance degradation suggests that the translation process introduces minimal computational overhead and can handle high-density IoT environments without becoming a bottleneck.

This behavior is particularly important in large-scale deployments, where edge gateways must process data streams from thousands of sensors simultaneously.

5.6 Resource Efficiency and Edge Deployment

The measured CPU utilization (11.8%) and memory footprint (~73 MB) indicate that the translator operates efficiently within the capabilities of modern edge devices, such as industrial gateways and embedded computing platforms. While this resource usage exceeds the limitations of Class 0 and Class 1 devices, it remains well within the capacity of Class 2 devices and above.

This validates the architectural decision to deploy the translator as an edge proxy, allowing constrained devices to participate in quantum workflows without incurring local computational overhead. By offloading processing to more capable nodes, the system preserves responsiveness while enabling scalability across heterogeneous IoT environments.

5.8 Deployment Feasibility and System Integration

The results demonstrate that, according to [20], the proposed translator can be effectively deployed within edge and fog computing environments, including 5G-enabled infrastructures. Its low latency, stable performance, and modest resource requirements make it well-suited for integration into existing IoT architectures.

Moreover, the deterministic processing model enhances resilience to variable network conditions, supporting reliable operation in dynamic industrial environments. This positions the translator as a practical intermediary layer for bridging classical IoT systems with emerging quantum computing platforms.

6. Limitations and Future Directions

While the Digit-Based Gate Mapping strategy provides a highly efficient and deterministic interface, it is important to note that the current performance evaluation was conducted within a logical simulation environment due to regional restrictions that currently limit access to physical quantum backends from our geographic location. Consequently, while the experimental results verify the logical consistency and edge-side translation latency of the system, they do not yet account for the specific noise profiles or queuing delays associated with physical Quantum Processing Units.

Furthermore, while the system is highly feasible for Class 2 edge devices, the measured 73.18 MB memory footprint remains a target for further optimization.

Future research will focus on:

- Optimizing the Qiskit runtime modules to lower the memory overhead for deployment on more constrained fog nodes.
- Exploring the scalability of this mapping strategy for complex data structures, such as high-dimensional vector data or encrypted telemetry.
- Developing adaptive digit-based mapping strategies to further minimize circuit complexity and improve translation efficiency.

7. Conclusion

This pioneering study successfully bridges the significant paradigm gap through the innovative translator architecture. By harnessing our strategy, the system delivers exceptionally low mean pipeline latency, near-perfect circuit generation success rates, and superior encoding fidelity over 1,024 execution shots per circuit.

The experimental results strongly affirm the computational prowess of our approach, demonstrating exceptionally low resource utilization on standard edge hardware. These remarkable characteristics unequivocally validate the deployment of the translator as a robust edge proxy, empowering even the most severely constrained IoT devices to seamlessly participate in quantum workflows—without necessitating any local computational upgrades.

Overall, the proposed architecture provides a scalable and practical pathway toward real-time quantum-integrated IoT systems. Future work may explore extending the mapping strategy to multi-qubit representations, improving encoding precision, and integrating with real quantum hardware backends to further validate performance under production conditions.

References

[1] R. E. Navas, L. Toutain, and K. Vijayasankar, “State of the art of IETF security related protocols for IoT,” Nov. 2016, Accessed: Jan. 2025. [Online]. Available: https://hal.science/hal-01522020

[2] H. Petersen, E. Baccelli, and M. Wählisch, “Interoperable Services on Constrained Devices in the Internet of Things,” Jun. 2014, Accessed: Sep. 2025. [Online]. Available: https://hal.inria.fr/hal-01058636

[3] A. Nahar, K. K. Mondal, D. Das, and R. Buyya, “qIoV: A Quantum-Driven Internet-of-Vehicles-Based Approach for Environmental Monitoring and Rapid Response Systems,” Mar. 2024, doi: 10.48550/arxiv.2403.18622.

[4] M. Bhatia and S. K. Sood, “Quantum Computing-Inspired Network Optimization for IoT Applications,” Mar. 2020, doi: 10.1109/jiot.2020.2979887.

[5] K. Wintersperger, W. Mauerer, and H. Safi, "QPU-System Co-Design for Quantum HPC Accelerators," Sep. 2022, doi: 10.5281/zenodo.7019315.

[6] C. B. Porciúncula, S. Beskow, É. S. Rocha, and J. C. Nobre, "Authentication and Authorization for Constrained Environments (ACE) com Framework OAuth e Protocolo CoAP," Oct. 2018, doi: 10.5902/2448190430934.

[7] R. Gutierrez, W. Villegas-Ch, and J. Govea, "Modular middleware for IoT: scalability, interoperability and energy efficiency in smart campus," Sep. 2025, doi: 10.3389/frcmn.2025.1672617.

[8] A. E. M. Elamin, "Bridging the Paradigm Gap: A Framework for Interoperability between Classical IoT and Quantum Computing Networks", IJC, vol. 57, no. 1, pp. 265–274, Apr. 2026, Accessed: Apr. 30, 2026. [Online]. Available: https://ijcjournal.org/InternationalJournal OfComputer/article/view/2530.

[9] J. L. Cacheiro *et al.* , "QMIO: A tightly integrated hybrid HPCQC system," May 2025, doi: 10.48550/arxiv.2505.19267.

[10] M. Chadha, J. John, and M. Gerndt, "Extending SLURM for Dynamic Resource-Aware Adaptive Batch Scheduling," 2020, doi: 10.48550/ARXIV.2009.08289.

[11] L. Reiher, B. Lampe, T. Woopen, R. van Kempen, T. Beemelmanns, and L. Eckstein, "Enabling Connectivity for Automated Mobility: A Novel MQTT-based Interface Evaluated in a 5G Case Study on Edge-Cloud Lidar Object Detection," Nov. 2022. doi: 10.1109/iceccme55909.2022.9987813.

[12] Z. Wen *et al.* , "Janus: Latency-Aware Traffic Scheduling for IoT Data Streaming in Edge Environments," Sep. 2023, doi: 10.1109/tsc.2023.3312131.

[13] H. T. Nguyen, P. Krishnan, D. Krishnaswamy, M. Usman, and R. Buyya, "Quantum Cloud Computing: A Review, Open Problems, and Future Directions," Apr. 2024, doi: 10.48550/arxiv.2404.11420.

[14] S. Tomar, R. K. Tripathi, and S. Kumar, "Comprehensive Survey of QML: From Data Analysis to Algorithmic Advancements," Jan. 2025, doi: 10.48550/arxiv.2501.09528.

[15] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration." Jan. 28, 2019. doi: 10.1145/3292674.

[16] R. Giambona, A. E. C. Redondi, and M. Cesana, "MQTT+," Oct. 2018, doi: 10.1145/3267129.3267135.

[17] M. Chadha, J. John, and M. Gerndt, "Extending SLURM for Dynamic Resource-Aware Adaptive Batch Scheduling," p. 223, Dec. 2020, doi: 10.1109/hipc50609.2020.00036.

[18] H. Derhamy, J. Eliasson, and J. Delsing, "IoT Interoperability—On-Demand and Low Latency Transparent Multiprotocol Translator," *IEEE Internet of Things Journal* , vol. 4, no. 5, p. 1754, Apr. 2017, doi: 10.1109/jiot.2017.2697718.

[19] H. Derhamy, "Towards Interoperable Industrial Internet of Things - An On-Demand Multi-Protocol Translator Service." 2016.

[20] D. M. K. Dave and B. K. Mittapally, "Data Integration and Interoperability in IOT: Challenges, Strategies and Future Direction." Jan. 2024.