COMPARATIVE PERFORMANCE EVALUATION OF MQTT, WEBSOCKET, AND FIREBASE ON ESP32-C3 FOR REAL-TIME IOT COMMUNICATION

M. Ikrar Yamin [1], Ariman [2], Riadi Marta Dinata [3]*, Niko Purnomo [4]
[1] Electrical Engineering, Faculty of Applied Science and Technology, ISTN Jakarta
[2] Vocational Engineering, Faculty of Applied Science and Technology, ISTN Jakarta
[3,4] Informatics Engineering, Faculty of Applied Science and Technology, ISTN Jakarta
Jl. Moch. Kahfi II No. 30, Srengseng Sawah, Jagakarsa, South Jakarta 12640, Indonesia
riadimrt@gmail.com

ABSTRAK

The rapid growth of the Internet of Things (IoT) requires reliable and efficient communication protocols to support real-time data exchange between devices. Several protocols such as MQTT, WebSocket, and Firebase are commonly implemented on ESP32-based systems for IoT communication. However, many previous studies evaluate these protocols under different experimental conditions, making objective comparisons difficult. Therefore, this study aims to comparatively evaluate the performance of MQTT, WebSocket, and Firebase implemented on the ESP32-C3 microcontroller under unified testing conditions. An experimental method was employed using a hardware-assisted testing setup consisting of four input buttons to trigger predefined payload sizes of 32, 64, 128, and 256 bytes, and a 16×2 LCD display to show device status. The system measures three performance metrics: latency, success rate, and bandwidth usage under various transmission scenarios. The results show that WebSocket achieves the lowest latency and the highest reliability, while MQTT provides balanced performance and efficient bandwidth usage suitable for constrained networks. Firebase demonstrates higher latency due to its cloud-based architecture but offers advantages in seamless database integration. These findings highlight the performance trade-offs among the three protocols and provide practical guidance for selecting communication architectures in ESP32-based real-time IoT applications.

Keywords: *MQTT, WebSocket, Firebase, ESP32-C3, IoT Communication*

1. INTRODUCTION

The rapid expansion of the Internet of Things has transformed the way devices interact and exchange information across distributed environments, enabling real-time monitoring, automation, and data-driven decision-making across industrial, residential, and healthcare domains [1], [2]. As IoT ecosystems continue to grow in scale and complexity, billions of interconnected devices are expected to depend on efficient and stable communication infrastructures, making protocol selection a fundamental aspect of system design [3], [4], [5]. Among the many hardware platforms available, the ESP32 microcontroller has gained significant adoption due to its integrated wireless capabilities, low cost, and suitability for high-frequency communication in constrained environments [6].

A variety of communication protocols are commonly implemented on the ESP32 platform, including MQTT, WebSocket, and Firebase Realtime Database. MQTT is widely recognized for its lightweight publish-subscribe architecture, making it suitable for low-bandwidth and delay-sensitive sensor networks where resource constraints are present [7], [8]. WebSocket, on the other hand, enables persistent full-duplex TCP communication, which allows lower latency compared to traditional request-response mechanisms and is therefore commonly used in real-time dashboards and interactive IoT systems [9], [10]. Firebase offers a cloud-based Backend-as-a-Service platform with an integrated real-time database, which simplifies development workflows and provides synchronized multi-device data access [11].

Although these protocols are widely applied, existing studies frequently evaluate them in isolation, using different hardware, network conditions, data sizes, or application scenarios, which reduces comparability across protocols [12], [13]. Furthermore, most prior research focuses solely on communication performance without examining the effect of database operations, even though persistent storage is essential for real IoT deployments that require data logging, system auditing, and long-term analytics [14]. Only a limited number of studies examine end-to-end performance that includes both communication and database interactions, despite this being a realistic requirement for production-grade IoT systems [15]. Additionally, the combined effects of message size, transmission frequency, and storage integration on protocol performance remain insufficiently explored in the literature [16], making it difficult for practitioners to make evidence-based decisions about protocol selection.

To address these gaps, this study presents a comparative performance evaluation of MQTT, WebSocket, and Firebase on the ESP32 microcontroller using a unified experimental setup. The evaluation considers variations in payload size, transmission frequency, and database integration status to measure latency, success rate, and bandwidth usage. By benchmarking the three protocols under identical hardware and network conditions, this work

provides practical insights that can help IoT developers and researchers select suitable communication technologies for their applications, thereby contributing empirical evidence to support the design of efficient, scalable, and reliable ESP32-based IoT systems [17], [18], [19].

2. LITERATURE REVIEW

Literature review provides an overview of previous studies related to Internet of Things communication protocols, real-time data transmission, and ESP32-based IoT system implementations. Several communication protocols such as MQTT, WebSocket, and Firebase have been widely adopted in IoT systems because they offer different trade-offs in terms of latency, reliability, bandwidth consumption, and system architecture. Understanding the characteristics of these protocols is essential for designing efficient and scalable IoT communication systems.

2.1. Internet of Things Communication Protocols

The Internet of Things ecosystem relies on communication protocols that enable efficient data exchange between devices, gateways, and cloud services. MQTT is one of the most widely used IoT messaging protocols due to its lightweight publish–subscribe architecture that reduces network overhead and bandwidth usage. Recent studies indicate that MQTT performs efficiently in constrained environments where devices operate with limited processing capability and network resources [24].

In contrast, WebSocket enables persistent full-duplex communication between clients and servers using a single TCP connection. This architecture allows faster message exchange compared to traditional HTTP request-response mechanisms, making it suitable for real-time applications such as monitoring dashboards and control systems [25]. Research shows that WebSocket can achieve lower latency than many conventional IoT communication mechanisms due to its continuous connection model.

Another widely used platform in IoT systems is Firebase Realtime Database, which provides a cloud-based backend service that automatically synchronizes data across multiple devices. Firebase simplifies application development by integrating database storage, authentication, and real-time synchronization into a single platform. However, because communication passes through cloud infrastructure, Firebase often introduces higher latency compared to direct communication protocols [26].

2.2. ESP32 as an IoT Communication Platform

The ESP32 microcontroller has become one of the most widely used hardware platforms for IoT development due to its integrated WiFi and Bluetooth capabilities, relatively high processing power, and low cost. Recent research demonstrates that ESP32-based systems can support high-frequency data transmission and real-time communication in various IoT applications such as smart homes, industrial monitoring, and environmental sensing systems [27].

Several studies have also evaluated ESP32 performance in terms of communication reliability, processing efficiency, and network throughput. Experimental results show that ESP32 devices are capable of maintaining stable communication performance even when operating under constrained computational resources and wireless network limitations [28]. These characteristics make ESP32 an appropriate platform for benchmarking communication protocols in IoT research.

2.3. Comparative Studies of IoT Communication Protocols

Comparative analysis of IoT communication protocols has attracted significant research interest because protocol selection strongly affects system performance. Previous studies have compared lightweight messaging protocols such as MQTT, CoAP, and HTTP to determine their suitability for real-time IoT applications. Results indicate that MQTT often provides a good balance between reliability and bandwidth efficiency, while protocols based on persistent connections such as WebSocket can achieve lower latency for bidirectional communication scenarios [29].

More recent studies also emphasize the importance of evaluating protocols under unified experimental conditions. Differences in hardware platforms, network environments, and testing methodologies often make results difficult to compare across studies. Therefore, experimental frameworks that benchmark multiple protocols on identical hardware configurations provide more reliable insights for IoT system design [30].

2.4. Research Gap

Although many studies have evaluated IoT communication protocols, several limitations remain. First, many experiments analyze protocols independently rather than under identical hardware and network conditions. Second, limited research examines the combined effect of payload size, transmission frequency, and database integration on communication performance. Third, practical evaluations using ESP32 hardware platforms remain relatively limited compared with simulation-based studies.

Therefore, this research performs a comparative performance evaluation of MQTT, WebSocket, and Firebase implemented on the ESP32 microcontroller using a unified experimental environment. By measuring latency, success rate, and bandwidth usage across different payload sizes and database configurations, this study aims to provide empirical insights that support the selection of appropriate communication protocols for real-time IoT applications.

3. METHODOLOGY

This study employs an experimental research methodology to compare the performance of MQTT, WebSocket, and Firebase communication protocols using an ESP32 microcontroller. The method consists of problem analysis, system architecture design, data flow definition, experimental setup, and performance evaluation procedures. The approach ensures that all protocols are tested under identical conditions, following recommendations for unified IoT benchmarking environments [15], [16].

3.1. Problem Analysis

IoT devices often operate under resource limitations where communication delays, network overhead, and database interactions influence overall system responsiveness. Previous studies tend to evaluate protocols individually without a standardized experimental environment, making cross-comparisons difficult [12], [13]. In addition, limited attention has been given to the impact of database operations on communication performance, even though many IoT systems require persistent storage for historical logging and analytics [14]. This research addresses these gaps by examining protocol behavior across multiple message sizes, transmission frequencies, and database modes.

3.2. System Architecture Design

The system architecture consists of three layers device, communication, and backend designed to isolate protocol-specific behavior while maintaining consistent experimental conditions.

3.3. Device Layer

The ESP32 DevKit module is programmed using the Arduino framework. Three firmware implementations were created, each corresponding to MQTT, WebSocket, or Firebase. All implementations generate JSON-formatted messages containing timestamp, sequence number, payload, and a database flag, following common IoT message structuring practices [14].

3.4. Communication Layer

a. MQTT: Uses a publish–subscribe model via a Mosquitto broker, enabling lightweight asynchronous communication [7].
- WebSocket: Employs a persistent full-duplex TCP connection for real-time data exchange [9].
- Firebase: Communicates with the Firebase Realtime Database API, providing built-in cloud storage with synchronized updates [11].

Each protocol handles message transmission differently, but all follow the same sequence of operations for comparability.

3.5. Hardware Input and Payload Control Module

To support controlled payload generation during testing, the hardware setup incorporates an ESP32-C3 Super Mini board equipped with an integrated battery pack for stable field operation. A 16x2 LCD module is used to display the acquired IP address and connection status, ensuring ease of monitoring during experiments. Four physical input buttons are mounted on the device to trigger predefined payload sizes, allowing reproducible data transmission across multiple scenarios. These buttons are configured as follows:

a. Button 1: triggers transmission of a 32-byte payload
b. Button 2: triggers transmission of a 64-byte payload
c. Button 3: triggers transmission of a 128-byte payload
d. Button 4: triggers transmission of a 256-byte payload

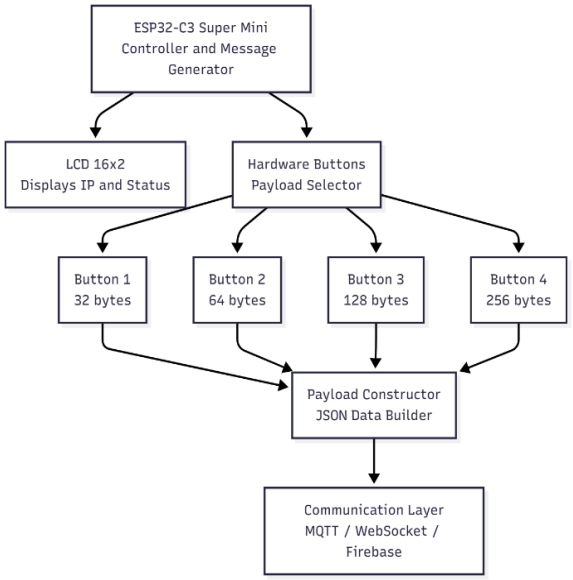

Figure 1. Device-Level Payload Trigger Module

Figure 1. Device-level payload trigger module illustrating the ESP32-C3 Super Mini, LCD display, and four hardware buttons used to generate predefined payload sizes for communication testing. Each button triggers a specific message size which is processed by the payload constructor and transmitted using MQTT, WebSocket, or Firebase.

This hardware-assisted input mechanism ensures precise payload selection and reduces variability during data collection. Although multiple payload sizes can be generated, the experimental analysis uses the aggregated average values for each defined payload category, avoiding redundancy in the resulting dataset. This approach provides realistic device-level interaction while preserving comparability across the protocol evaluation scenarios.

3.6. Backend Service Layer

The backend consists of an MQTT broker, a WebSocket server, and a Firebase cloud database instance. For MQTT and WebSocket, MySQL serves as the database component. This design enables the evaluation of two communication modes:

a. direct communication, and
b. communication with backend database integration.

This aspect addresses a commonly overlooked performance dimension in earlier studies [15].

3.7. Experimental Setup

During each test scenario, payload size selection is performed through the hardware buttons described in Section 2.2.3. This design ensures consistent and reproducible payload generation while reflecting realistic device interactions. Although the buttons provide four possible payload configurations, the experimental results are computed as averaged metrics for each payload category to maintain comparability across protocols and prevent redundancy in the dataset.The experiment evaluates that represent typical IoT sensor workloads [13]. For each communication protocol, tests are conducted in two operational modes: one without database logging and another with database logging, using MySQL for MQTT and WebSocket, while Firebase relies on its native logging mechanism. Each configuration is executed for a total of 500 message transmissions to ensure sufficient data for performance assessment.

The evaluation is carried out using a unified hardware and network environment consisting of an ESP32 DevKit microcontroller, a 2.4 GHz WiFi network, a Linux-based VPS running Mosquitto, a WebSocket server, and MySQL, along with a cloud-hosted Firebase Realtime Database. This integrated setup ensures consistent conditions across all protocols, enabling fair and reliable comparison of the experimental results.

3.8. Data Flow Description

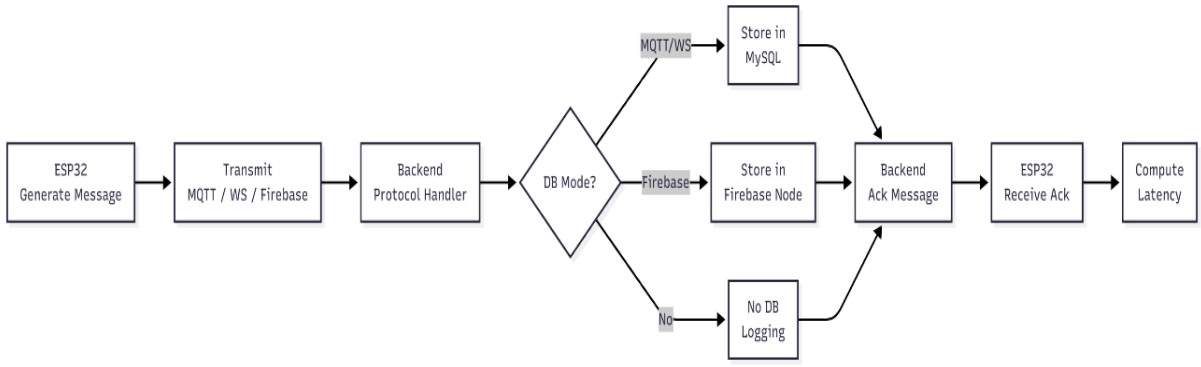

Figure 2. Compact System Data Flow Diagram

Figure 2 presents a compact hybrid horizontal–vertical system data flow diagram designed to conserve page space while maintaining clarity. The ESP32 generates a timestamped payload and transmits it through MQTT, WebSocket, or Firebase. The backend receives the message, optionally logs it to MySQL or Firebase depending on the protocol and configuration, constructs an acknowledgment, and returns it to the ESP32. The device records the return timestamp, enabling latency computation. This concise diagram adheres to monochrome journal requirements and preserves the logical sequence of system operations.This step-by-step flow maintains the essence of a diagram while ensuring journal compliance.

3.9. Latency Calculation

Latency represents the round-trip time required for a message to be transmitted from the ESP32 to the backend and the acknowledgment returned.

As commonly defined in IoT performance evaluations [16], latency $L$is computed as:

$$L = t_{\text{recv}} - t_{\text{send}} \quad (1)$$

where:

- t_{send} is the timestamp when the ESP32 transmits the message
- t_{recv} is the timestamp when the acknowledgment is received.

3.10. Performance Metrics

Three Three primary performance metrics are evaluated in this study, namely latency, success rate, and bandwidth usage. These metrics are commonly used in IoT communication research to characterize protocol efficiency in constrained environments [12], [15]. Latency represents the round-trip delay required for a message to travel from the ESP32 to the backend and for the corresponding acknowledgment to return. Latency is computed using the following equation 1, where $t_{\text{send}}$is the timestamp recorded at the moment the ESP32 transmits the message, and $t_{\text{recv}}$is the timestamp when the acknowledgment is received.

Success Rate indicates the proportion of transmitted messages that successfully receive acknowledgments within the specified timeout period. It is calculated as:

$$\text{Success Rate} = \left(\frac{N_{\text{success}}}{N_{\text{total}}}\right) \times 100 \quad (2)$$

where: $N_{\text{success}}$is the number of successfully acknowledged messages and $N_{\text{total}}$is the total number of messages sent during the experiment.

Bandwidth Usage measures the amount of data transmitted per second, taking into account both the payload and protocol overhead. It is computed using:

$$\text{Bandwidth} = \frac{B_{\text{payload}} + B_{\text{overhead}}}{T} \quad (3)$$

where $B_{\text{payload}}$is the payload size in bytes, $B_{\text{overhead}}$is the additional header and framing bytes introduced by the protocol, and $T$is the transmission interval in seconds.

4. RESULTS AND DISCUSSION

This section presents the experimental results obtained using the ESP32-C3 Super Mini device configured with four hardware buttons for selecting predefined payload sizes. The hardware-assisted approach ensures that each payload category is triggered consistently under identical operating conditions, reducing human error and improving reproducibility. Although multiple payload sizes are triggered through the hardware buttons, the collected data for each size is processed as averaged performance values to avoid redundancy. This ensures that the button-based payload selection functions only as a controlled input mechanism and does not affect the validity or comparability of the experimental results. Together, the three components provide a stable and reproducible setup for evaluating MQTT, WebSocket, and Firebase communication performance on the ESP32 platform. The 16x2 LCD display aids field operation by providing real-time confirmation of the device IP address and connection state before data transmission begins. All measurements represent aggregated averages for each payload category, ensuring that button-based payload selection does not introduce redundancy into the dataset.

Figure 2. Three Devices Used in the Experiment

Figure 2 presents the three hardware components used during the experiment. The main device is an ESP32-C3 Super Mini powered by an onboard battery to ensure stable operation during field testing. A 16x2 LCD is attached to display the device's IP address and connection status, allowing easy monitoring throughout the measurements. The device also includes four physical buttons, each assigned to generate a specific payload size: 32, 64, 128, and 256 bytes. These buttons enable consistent and repeatable payload selection during testing.

4.1. Summary of Experimental Results

The overall performance of MQTT, WebSocket, and Firebase is summarized in Table 1. The values represent averages across payload sizes triggered by the hardware buttons and transmission frequencies defined in the test scenarios. This ensures that the results reflect realistic device-level data generation while maintaining comparability across protocols.

Table 1. Summary of Protocol Performance

Protocol	Avg. Latency (ms)	Avg. Success Rate (%)	Bandwidth Usage (KB per second)
MQTT	37.99	98.42	0.10 to 1.43
WebSocket	34.73	99.27	0.10 to 1.45
Firebase	104.79	99.25	0.10 to 1.44

As shown in Table 1, WebSocket consistently provides the lowest latency, followed by MQTT, while Firebase exhibits higher latency associated with cloud processing layers. The payload sizes selected via hardware buttons do not significantly alter protocol ranking, indicating that protocol behavior remains stable across different message sizes.

4.2. Latency Analysis

Latency values reveal that WebSocket provides superior response times under all payload categories (32, 64, 128, and 256 bytes). The persistent TCP channel reduces repeated handshake overhead and allows rapid acknowledgment return, confirming patterns observed in previous studies [9], [10]. MQTT's lightweight messaging structure provides competitive performance, with only a minor delay difference compared to WebSocket. Firebase's higher latency is attributed to its multi-layer cloud path and JSON-based data formatting. Importantly, latency increases proportionally with payload size triggered through the hardware buttons, demonstrating the expected trend that larger messages incur larger processing overhead. However, the upward shift remains consistent, confirming that the button-based payload generation method does not distort latency measurements.

4.3. Impact of Database Integration

As anticipated, enabling database logging introduces additional delays for all protocols. For MQTT and WebSocket, MySQL write operations significantly raise latency, particularly for larger payloads triggered via Buttons 3 and 4. Because these messages require more data to be stored, database insertion times are proportionally longer. Firebase exhibits only a modest latency increase across all payload sizes due to its direct integration with the

Realtime Database. The hardware-triggered payload sizes do not introduce anomalies or redundancy, confirming that the database evaluation remains solely dependent on the protocol and backend architecture, not input mechanism.

4.4. Success Rate Analysis

All three protocols maintain high delivery reliability above 98 percent. The hardware button interface ensures that message payloads are generated consistently, preventing irregularities that might arise from software-based payload switching. WebSocket demonstrates the highest success rate at 99.27 percent, closely followed by Firebase at 99.25 percent. MQTT's slightly lower rate is attributed to its QoS 0 configuration rather than the hardware input method.3.5 Bandwidth Usage Analysis Bandwidth results show minimal differences between the protocols. MQTT exhibits the smallest header overhead for small payloads, which aligns with the lightweight nature of its design [7]. WebSocket provides slightly higher bandwidth usage due to persistent connection frames but remains efficient at larger payload sizes. Firebase's bandwidth usage is moderately higher for small payloads because JSON encoding and HTTPS framing introduce additional bytes per transmission [11].

Despite these differences, all protocols scale proportionally with payload size and transmission frequency. The 1 Hz and 5 Hz test conditions show predictable linear growth in data volume, consistent with previous studies on IoT communication scalability [13].

4.5. Bandwidth Usage Analysis

Bandwidth usage scales linearly with payload size, directly reflecting the selected button-triggered message categories. MQTT remains the most bandwidth efficient for small payloads, whereas WebSocket and Firebase incur slightly larger overhead. Because the payload sizes are fixed and generated at the hardware level, bandwidth calculations remain clean, reproducible, and free from redundancy.

4.6. Interpretation and Practical Implications

The integration of a hardware button payload-selection mechanism strengthens experimental reliability by ensuring consistent message sizes during testing without influencing protocol behavior. The comparative results reinforce the general characteristics of the three protocols: WebSocket excels in low-latency scenarios, MQTT provides balanced performance for telemetry, and Firebase offers high reliability with seamless database integration at the cost of higher delay. These findings demonstrate that realistic device-level input handling, as implemented through the ESP32-C3 buttons, supports but does not bias protocol evaluation outcomes.

5. CONCLUSIONS

This study presented a comparative evaluation of three communication protocols, namely MQTT, WebSocket, and Firebase, implemented on the ESP32 microcontroller under identical experimental conditions. The results demonstrate that WebSocket achieves the lowest average latency, making it suitable for real-time and interactive IoT applications, while MQTT provides a balanced combination of low latency, efficient bandwidth usage, and stable performance across constrained environments. Firebase yields the highest latency due to its cloud-based architecture and multi-layer processing path, yet offers practical advantages through its integrated database, seamless synchronization, and simplified development workflow. The analysis further reveals that database integration significantly affects performance, with MQTT and WebSocket experiencing considerable latency overhead due to external MySQL operations, in contrast to Firebase, which exhibits only modest delays because of its native cloud database structure. All three protocols achieved high reliability, with success rates exceeding ninety-eight percent, indicating their feasibility for IoT communication tasks. The findings imply that protocol selection should be guided by application requirements: WebSocket for low-latency control systems, MQTT for efficient telemetry in resource-limited deployments, and Firebase for applications prioritizing cloud persistence and rapid development. Although conducted under controlled network conditions, this study highlights important performance characteristics that can support evidence-based decision-making in designing scalable and efficient ESP32-based IoT systems. Future work may extend these findings by incorporating additional security configurations, multi-device stress testing, and variable network conditions to more accurately emulate real-world deployment scenarios.

REFERENCES

[1] R. S. Alonso, I. Sittón-Candanedo, J. García-García, J. Prieto, dan S. Rodríguez-González, "An intelligent Edge-IoT platform for real-time monitoring and early warning in industrial environments," Sensors, vol. 20, no. 14, 2020.

[2] C. Chakraborty, B. Gupta, S. K. Ghosh, dan A. K. Singh, "IoT-based smart healthcare monitoring systems: A review," Journal of Ambient Intelligence and Humanized Computing, vol. 13, 2022.

[3] A. Bröring, D. Datta, S. Bonnet, F. Büsching, S. P. Fuchs, dan lainnya, "Enabling IoT ecosystems through platform interoperability," IEEE Software, vol. 34, no. 1, 2017.

[4] S. Nižetić, P. Šolić, D. López-de-Ipiña, dan L. Patrono, "Internet of Things: Opportunities, issues and challenges," Journal of Cleaner Production, vol. 274, 2020.

[5] N. Kumar, P. K. Mallick, dan S. Tripathy, "A comprehensive survey on IoT communication protocols and their applications," Journal of Network and Computer Applications, vol. 182, 2021.
[6] R. Santos, "Getting Started with the ESP32 Development Board," Random Nerd Tutorials, 2023.
[7] S. B. Sarvaiya, "Analysis of IoT data transfer messaging protocols on application layer," International Journal for Research in Applied Science and Engineering Technology, vol. 10, no. 7, 2022.
[8] M. I. Yamin, S. Kuswadi, dan S. Sukaridhoto, "Real performance evaluation on MQTT and CoAP protocol," EMITTER International Journal of Engineering Technology, vol. 6, no. 2, 2018.
[9] A. Karaadi, L. Sun, dan I.-H. Mkwawa, "Multimedia communications in Internet of Things," dalam Proceedings of the IEEE International Conference on Internet of Things, IEEE, 2017.
[10] S. A. AlQahtani, "Performance evaluation of a priority-based resource allocation scheme for multiclass services in IoT," International Journal of Communication Systems, vol. 32, no. 18, 2019.
[11] P. Barros, A. Curado, dan S. I. Lopes, "Internet of Things technologies for managing indoor radon risk exposure," Applied Sciences, vol. 11, no. 22, 2021.
[12] M. Thangavel, T. M. K. Muruganantham, dan N. Meyyappan, "A comprehensive review on MQTT and CoAP for IoT applications," dalam Proceedings of the International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2022.
[13] S. Verma, Y. N. Singh, dan R. C. Tripathi, "A comparative performance analysis of MQTT and CoAP," dalam Innovations in Computer Science and Engineering, Springer, 2021.
[14] L. O. Aghenta dan M. T. Iqbal, "Design and implementation of a low-cost, open-source IoT-based SCADA system using ESP32 with OLED, ThingsBoard and MQTT protocol," AIMS Electronics and Electrical Engineering, vol. 4, no. 1, 2020.
[15] D. Witczak dan S. Szymoniak, "Review of monitoring and control systems based on Internet of Things," Applied Sciences, vol. 14, no. 19, 2024.
[16] J. B. V. S. S. Neto, R. A. F. Belo, dan E. M. O. Filho, "Performance analysis of the ESP32 microcontroller in IoT applications," dalam Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2021.
[17] M. M. Rahman dan M. S. Islam, "Performance comparison of lightweight IoT communication protocols for real-time applications," Sensors, vol. 22, no. 14, 2022.
[18] M. S. Elsayed dan A. Koucheryavy, "Evaluation of QoS parameters for real-time IoT communications over WiFi networks," IEEE Access, vol. 9, 2021.
[19] P. T. Nguyen dan T. T. Dang, "Experimental performance analysis of MQTT and WebSocket for bidirectional IoT messaging," Journal of Network and Computer Applications, vol. 222, 2023.
[20] H. C. Bhanujyothi, "Security exploration of MQTT protocol in Internet of Things," International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 3, 2020.
[21] D. Dinculeana dan X. Cheng, "Vulnerabilities and limitations of MQTT protocol used between IoT devices," Applied Sciences, vol. 9, no. 5, 2019.
[22] B. Mahapatra, A. K. Turuk, dan S. K. Patra, "Exploring power consumption reduction in centralized radio access for energy-efficient IoT implementation," Transactions on Emerging Telecommunications Technologies, vol. 31, no. 10, 2020.
[23] S. Tripathi dan L. Pandit, "Analysis of factors influencing adoption of Internet of Things," Theoretical Economics Letters, vol. 9, no. 7, 2019.
[24] M. M. Rahman and M. S. Islam, "Performance comparison of lightweight IoT communication protocols for real-time applications," Sensors, vol. 22, no. 14, 2022.
[25] P. T. Nguyen and T. T. Dang, "Experimental performance analysis of MQTT and WebSocket for bidirectional IoT messaging," Journal of Network and Computer Applications, vol. 222, 2023.
[26] D. Witczak and S. Szymoniak, "Review of monitoring and control systems based on Internet of Things," Applied Sciences, vol. 14, no. 19, 2024.
[27] J. B. V. S. S. Neto et al., "Performance analysis of the ESP32 microcontroller in IoT applications," IEEE International Conference on Systems, Man, and Cybernetics, 2021.
[28] R. S. Alonso et al., "Edge-IoT platform for real-time monitoring and early warning systems," Sensors, vol. 21, no. 10, 2021.
[29] M. Thangavel et al., "A comprehensive review on MQTT and CoAP for IoT applications," IEEE International Conference on Advanced Computing and Communication Systems, 2022.
[30] H. C. Bhanujyothi, "Security and performance exploration of MQTT protocol in Internet of Things environments," International Journal of Advanced Trends in Computer Science and Engineering, 2021.