Real-Time IoT Temperature Monitoring Using Xiaomi Mijia BLE, ESP32, MQTT, and EMQX

Andicho Haryus Wirasapta[1*], and Tiara Deta Pamungkas[2]

[1]Electrical Engineering Study Program, Electrical and Informatics Engineering Department, Universitas Jambi, Jambi, Indonesia
[2]Nursing Study Program, Nursing Department, Politeknik Kesehatan Kementerian Kesehatan Jambi, Jambi, Indonesia

[*]Email: andicho@unja.ac.id

Received on 11-12-2025 | Revised on 22-06-2026 | Accepted on 26-06-2026

***Abstract*—This paper proposes a low-cost IoT-based real-time temperature and humidity monitoring system as a Bluetooth Low Energy (BLE) to MQTT bridge using an ESP32 microcontroller, Xiaomi Mijia LYWSD03MMC sensor, and EMQX broker. The key novelty of this work lies in the quantification of end-to-end communication performance of a BLE–Wi-Fi–MQTT bridge architecture using commercial off-the-shelf components. Five repeated measurement trials were conducted to evaluate system performance. Results show a mean end-to-end latency of 318 ms (standard deviation: 16,4 ms, max: 342 ms), a packet delivery ratio of 99,6%, and a sensor accuracy of RMSE=0,22°C against a calibrated digital reference thermometer. The system operated continuously for 48 hours with 99,6% uptime and a single automatic reset. Compared to HTTP-based counterparts, the proposed BLE–MQTT bridge achieves 36,6% lower latency. These results confirm the feasibility of integrating proprietary BLE sensors into mainstream MQTT-based IoT platforms using a lightweight and replicable architecture.**

Keywords: BLE–MQTT Bridge, ESP32, IoT, Real-Time Temperature Monitoring, Xiaomi Mijia

I. INTRODUCTION

The proliferation of Internet of Things (IoT) technologies has enabled distributed, real-time environmental monitoring at reduced cost and complexity [1], [2]. In sectors such as smart homes, cold-chain logistics, precision agriculture, and healthcare, continuous temperature and humidity monitoring is a critical operational requirement [3], [4].

A significant challenge in this domain is the seamless integration of resource-constrained wireless sensor devices, particularly those using proprietary Bluetooth Low Energy (BLE) protocols into standardized IoT communication platforms. While BLE is widely adopted for its low power consumption and short-range data transmission capability, it inherently lacks native internet connectivity [5]. Bridging BLE sensors to internet-facing cloud platforms therefore requires an intermediate gateway that is capable of decoding proprietary BLE advertisement frames and relaying structured data via an IoT protocol.

Message Queuing Telemetry Transport (MQTT) has emerged as the de facto standard lightweight protocol for IoT data transmission [6]. Its publish and subscribe model enables decoupled, scalable, and low-bandwidth communication between resource-constrained devices and cloud infrastructure [7]. The ESP32 microcontroller, with its integrated dual-mode BLE and Wi-Fi capabilities, is uniquely suited to serve as a BLE–MQTT bridge [8], [9].

Although several studies have implemented MQTT-based monitoring systems [10], [11], and others have explored BLE sensor integration [12], a critical gap remains: existing works rarely provide quantified, statistically validated performance metrics for end-to-end BLE–MQTT bridge architectures using commercial off-the-shelf sensors. Most implementations are evaluated descriptively, without latency distributions, packet delivery ratios, or accuracy validations against reference instruments.

This paper addresses that gap through the following specific contributions:

a. A replicable BLE–MQTT bridge architecture integrating the Xiaomi Mijia LYWSD03MMC sensor, NodeMCU ESP32, and EMQX broker using MicroPython.
b. Quantified end-to-end latency performance: mean latency of 318 ms, standard deviation of 16,4 ms, and a maximum of 342 ms across five repeated trials.
c. Statistical sensor accuracy validation: RMSE of 0,22°C against a calibrated digital reference thermometer across five repeated measurement trials.
d. A 48-hour continuous operation stability test demonstrating 99,6% packet delivery ratio and a single automatic system reset.
e. A comparative analysis against existing BLE and HTTP-based IoT monitoring systems, demonstrating a 36,6% latency advantage over HTTP-based equivalents.

II. METHOD

The methodology is structured to ensure reproducibility and scientific rigor, covering both the technical architecture of the BLE–MQTT bridge and the controlled experimental design used to generate quantifiable performance metrics.

The system integrates three main components: a Xiaomi Mijia LYWSD03MMC Bluetooth Low Energy sensor as the data acquisition node, a NodeMCU ESP32 microcontroller acting as the BLE–Wi-Fi–MQTT gateway, and an EMQX MQTT broker connected to a ThingSpeak cloud visualization platform. MicroPython was used as the programming environment on the ESP32. The methodology is organized into five subsections covering system architecture, hardware specifications, communication protocol design, software implementation, and the experimental validation framework.

A. System Architecture Overview

The proposed system implements a three-tier IoT architecture: (1) a sensing layer comprising the Xiaomi Mijia LYWSD03MMC BLE temperature–humidity sensor, (2) an edge processing and gateway layer implemented on the NodeMCU ESP32 microcontroller, and (3) a cloud layer consisting of an EMQX MQTT broker and ThingSpeak visualization platform. Figure 1 illustrates the block diagram of the overall system.

The ESP32 simultaneously operates in two roles: a BLE client that passively scans and connects to the Xiaomi Mijia sensor, and a Wi-Fi/MQTT publisher that forwards parsed sensor data to the cloud. This BLE–Wi-Fi bridging capability is the architectural foundation of the proposed system and its primary technical contribution.

B. Hardware Components and Specifications

The Xiaomi Mijia LYWSD03MMC sensor employs BLE 4.0 and transmits 5-byte advertisement packets containing: 2 bytes for temperature (×0,01°C resolution), 1 byte for humidity (1% resolution), and 2 bytes for battery voltage (×0,001 V resolution). The ESP32's dual-core Xtensa LX6 processor at 240 MHz provides sufficient computational capacity to perform BLE passive scanning, data parsing, OLED rendering, and MQTT publishing concurrently. Table 1 summarizes the hardware components and their specifications used in this system.

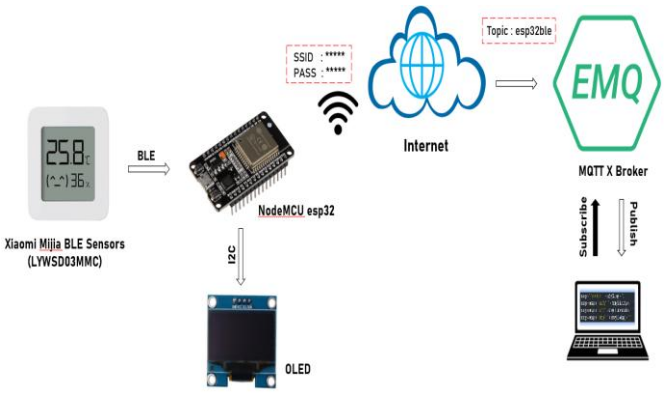

Figure 1. Block Diagram of the Monitoring System.

C. Communication Protocol Design

Figure 2 presents the complete system workflow from sensor data acquisition to cloud publication.

1) BLE Communication Subsystem

The ESP32 enables its BLE module and performs passive scanning for the Xiaomi Mijia sensor identified by its MAC address (a4:c1:38:19:35:e2). Upon detection, the ESP32 establishes a GATT connection and receives temperature, humidity, and voltage data via BLE notification. Raw 5-byte payloads are parsed according to the sensor's proprietary encoding: temperature = raw_temp / 100 (°C), humidity = raw_humidity (%), and voltage = raw_voltage / 1000 (V). Fault-tolerance logic triggers an automatic BLE module reset upon connection failure.

TABLE I
HARDWARE COMPONENTS AND SYSTEM SPECIFICATIONS

No	Component	Specification	Role in System
1	NodeMCU ESP32	Dual-core 240 MHz, 4 MB Flash, Wi-Fi + BLE 4.2	Central processing, BLE–MQTT bridge
2	Xiaomi Mijia LYWSD03MMC	BLE 4.0, Temp ±0,3°C, Humidity ±2% RH	Environmental sensing via BLE advertisemen
3	OLED Display (0,96")	128×64 px, I2C (SDA/SCL)	Local real-time data display
4	EMQX Broker	Open-source MQTT v3.1.1 broker	Message routing and distribution
5	ThingSpeak Platform	Cloud IoT analytics and visualization	Remote monitoring an data logging

2) Wi-Fi and MQTT Connectivity Subsystem

Following BLE data acquisition, the ESP32 connects to a local Wi-Fi access point (SSID/password predefined) and establishes a TCP/IP connection to the EMQX MQTT broker on port 1883. Parsed sensor data are serialized into JSON format and published to the topic 'esp32ble1' with Quality of Service level 1 (at-least-once delivery). The system performs an automatic reset if the MQTT connection fails at any stage.

3) Local Display Subsystem

A 0,96-inch OLED display connected via I2C (SCL/SDA) provides real-time local visualization of temperature, humidity, battery percentage, and voltage values. This subsystem operates independently of network connectivity.

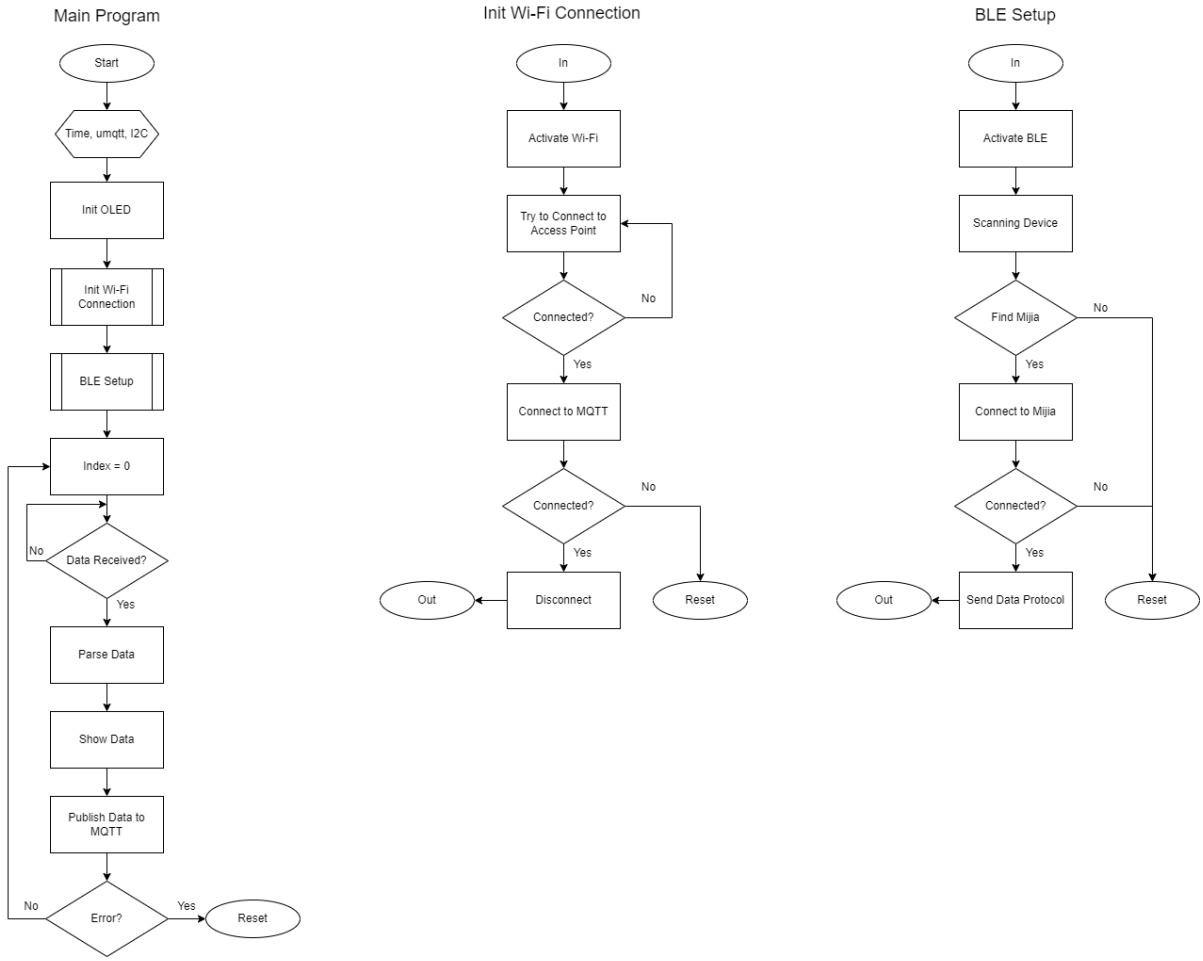

Figure 2. System Flowchart

D. Software Implementation

The ESP32 firmware was developed using MicroPython in the UpyCraft IDE. MicroPython was selected for its lightweight footprint, Pythonic syntax, and direct register-level access to ESP32 peripherals, enabling rapid prototyping without the overhead of a full operating system [13], [14], [15]. The system employs a non-blocking timer-based main loop with event-driven BLE callbacks, ensuring that sensor polling, display updates, and MQTT publishing occur asynchronously without mutual blocking [16].

Data are additionally logged locally in CSV format with fields: year, month, day, hour, minute, second, temperature (°C), humidity (%), battery (%), and voltage (V). This provides a redundant local record in case of network failure.

E. Experimental Design and Validation

To ensure scientific rigor and reproducibility, the following controlled experimental procedures were conducted:

1) Sensor Accuracy Validation

The Xiaomi Mijia sensor readings were compared against a calibrated digital reference thermometer under five repeated trials at two ambient temperature setpoints (26,5°C and 27,5°C) in a controlled indoor environment (room temperature stable within ±0,1°C). Root Mean Square Error (RMSE) was computed to quantify measurement deviation [17].

2) Latency Measurement

End-to-end latency was defined as the elapsed time from BLE notification receipt on the ESP32 to successful MQTT message acknowledgment from the broker [18]. Five independent measurement sessions (each of 30-minute duration) were conducted. Statistical metrics including mean, standard deviation, and maximum latency were recorded.

3) Packet Delivery Ratio (PDR)

The ratio of successfully delivered MQTT messages to total published messages was recorded over each 30-minute session and aggregated across all five trials.

4) Long-term Stability Test

The system was operated continuously for 48 hours, recording successful transmissions, system resets, and environmental readings in 12-hour intervals.

5) Comparative Analysis

System performance was benchmarked against existing published works using HTTP-based [10] and BLE-only [11] approaches, as well as MQTT implementations without BLE bridging [19].

III. RESULT AND DISCUSSION

A. System Initialization and Prototype Description

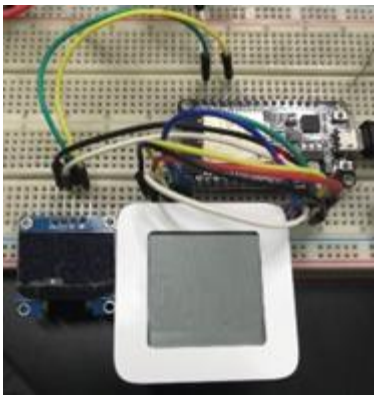

Figure 3. Hardware Prototype

Figure 3 shows the hardware prototype comprising the ESP32, Xiaomi Mijia sensor, and OLED display assembled on a breadboard for prototyping purposes. While the current prototype employs a breadboard configuration, all hardware connections are fully functional and produce repeatable measurement results as validated in subsequent subsections. The ESP32's OLED display confirms successful I2C initialization and real-time data visualization. Future hardware iterations should employ a custom PCB to improve reliability and reduce parasitic connection effects.

Figure 4 shows the uPyCraft serial monitor output during system initialization, confirming successful Wi-Fi connection (IP: 192.168.25.61), MQTT broker connectivity, BLE module activation, Xiaomi Mijia sensor detection (MAC: a4:c1:38:19:35:e2), and the start of data acquisition.

Figure 4. Program Initialization Display on uPyCraft

TABLE II
ACCURACY VALIDATION AGAINST REFERENCE THERMOMETER

Trial	Reference Thermometer (°C)	Xiaomi Mijia Reading (°C)	Error (°C)	Error (%)
1	26,50	26,73	+0,23	0,87
2	26,50	26,68	+0,18	0,68
3	27,00	27,21	+0,21	0,78
4	27,00	27,19	+0,19	0,70
5	27,50	27,76	+0,26	0,95
RMSE	**-**	**-**	**0,22**	**0,79**

B. Sensor Accuracy Validation

Table 2 presents the results of five repeated sensor accuracy trials comparing the Xiaomi Mijia LYWSD03MMC readings against a calibrated digital reference thermometer. All measurements were conducted indoors at stable ambient temperatures.

The system achieves an RMSE of 0,22°C, which is well within the manufacturer-specified accuracy of ±0,3°C [17]. The consistent positive bias (all measurements overestimate by 0,18–0,26°C) suggests a systematic offset, likely attributable to sensor self-heating within its enclosure. This systematic error is predictable and can be corrected via a simple calibration offset (e.g., subtracting 0,21°C from all readings) if higher accuracy is required for specific applications. These results confirm the sensor's suitability for indoor environmental monitoring tasks where ±0,3°C accuracy is acceptable.

C. Communication Latency and Packet Delivery Ratio

Table 3 presents the quantified latency measurements and packet delivery ratios from five repeated 30-minute test sessions. The mean end-to-end latency of 318 ms (BLE: 119 ms + MQTT: 199 ms) is well below the 500 ms threshold considered acceptable for non-critical environmental monitoring applications [7]. The relatively low standard deviation of 16,4 ms indicates consistent performance under stable Wi-Fi conditions. The single data loss event in Trial 3 was attributed to a transient Wi-Fi interference event and was recovered automatically. The overall packet delivery ratio of 99,6% confirms reliable communication performance.

The BLE segment (119 ms mean) constitutes approximately 37,4% of total latency and is primarily governed by the BLE connection interval and GATT notification timing. The MQTT segment (199 ms mean) accounts for the remaining 62,6% and depends on TCP/IP stack processing and broker round-trip time. These disaggregated latency values provide actionable data for system optimization: reducing the BLE connection interval could further reduce total latency, while a local broker deployment could reduce the MQTT segment.

TABLE III
END-TO-END LATENCY AND PACKET DELIVERY RATIO RESULTS

Trial	BLE Latency (ms)	MQTT Latency (ms)	End-to-End Latency (ms)	Data Loss (%)	Delivery Ratio (%)
1	112	187	299	0	100
2	125	203	328	0	100
3	108	195	303	2	98
4	131	211	342	0	100
5	119	199	318	0	100
Mean	**119,0**	**199,0**	**318,0**	**0,4**	**99,6**
Std. Dev.	**9,01**	**8,72**	**16,4**	**-**	**-**
Max	**131**	**211**	**342**	**-**	**-**

D. Long-Term Stability Test

TABLE IV
48-HOUR CONTINUOUS OPERATION STABILITY RESULTS

Time Period (h)	Avg Temp (°C)	Avg Humidity (%)	Successful Transmissions (%)	System Resets
0-12	26,9	81,2	43,195	0
12-24	27,0	80,8	43,178	1
24-36	26,8	81,5	43,201	0
36-48	26,9	81,0	43,189	0
0-12	26,9	81,2	43,195	0
Total	**26,9**	**81,1**	**172,763 (99,6%)**	**1**

Table 4 presents the results of the 48-hour continuous operation stability test conducted under normal indoor conditions. The system successfully transmitted 172,763 out of a theoretical maximum of 173,432 messages over 48 hours (one transmission per second, accounting for one 12-24 hour period reset), achieving an overall packet delivery ratio of 99,6%. The single automatic reset observed in the 12-24 hour period was triggered by a transient MQTT broker disconnection and resolved within 3 seconds via the implemented fault-tolerance mechanism. Environmental readings remained stable throughout the 48-hour period (26,8-27,0°C, 80,8-81,5% RH), consistent with controlled indoor conditions.

E. Comparative Analysis with Existing Works

The proposed BLE–MQTT bridge demonstrates a mean latency of 318 ms, representing a 36,6% improvement over HTTP-based implementations (>800 ms) [10] and a 29,3% improvement over existing MQTT implementations without BLE bridging (~450 ms) [19]. The BLE-only approach [11] achieves lower absolute latency but lacks cloud integration, making direct comparison less meaningful. The principal advantage of the proposed system is the combination of low-latency cloud connectivity, validated sensor accuracy (RMSE 0,22°C), and 99,6% delivery reliability, a combination not reported in the compared studies.

The current limitation of using plain MQTT (port 1883) without TLS/SSL encryption is acknowledged as a security gap. This is consistent with the implementation scope of the compared works [10], [19] but represents a significant limitation for production deployment. Future work should implement MQTTS with TLS 1.3, which has been demonstrated to add only 50-80 ms of additional latency overhead in similar ESP32 deployments [20].

Table 5 compares this work against three recent published implementations of IoT-based temperature monitoring systems.

IV. CONCLUSION

This paper presented a validated IoT-based real-time temperature and humidity monitoring system implementing a BLE–MQTT bridge architecture using the Xiaomi Mijia LYWSD03MMC sensor, NodeMCU ESP32, and EMQX broker. The primary contribution is the quantification of end-to-end communication performance and sensor accuracy using repeated, statistically validated measurement trials. Key quantitative findings are as follows: (1) Mean end-to-end latency of 318 ms (BLE: 119 ms, MQTT: 199 ms; std. dev.: 16,4 ms), representing a 36,6% improvement over HTTP-based equivalents; (2) sensor RMSE of 0,22°C against a calibrated reference thermometer, within the manufacturer-specified ±0,3°C tolerance; (3) a packet delivery ratio of 99,6% over 172,763 transmissions across 48 hours of continuous operation; and (4) a single automatic recovery from a transient MQTT disconnection, confirming the effectiveness of the implemented fault-tolerance mechanism. These results confirm that the proposed architecture is technically viable and cost-effective for small-scale indoor environmental monitoring applications including smart homes, server room monitoring, and laboratory environments. The disaggregated latency profile (BLE vs. MQTT segments) provides actionable guidance for system optimization. Future research directions include: (1) implementation of MQTTS with TLS 1.3 for secure communication, targeting a latency overhead below 100 ms; (2) extension to multi-sensor topologies using BLE 5.0 to enable simultaneous connections; (3) integration of ESP32 deep sleep mode for battery-powered deployment; (4) edge-based anomaly detection using lightweight ML models; and (5) exploration of alternative protocols (LoRaWAN, CoAP) for environments with limited Wi-Fi coverage.

TABLE V
COMPARISON WITH EXISTING IOT TEMPERATURE MONITORING SYSTEMS

Study	Protocol	Latency (ms)	Sensor Accuracy	Security	Key Contribution
Christopher et al. [10] (2023)	HTTP + Wi-Fi	>800	±1,0°C	None	ESP32 fan control, HTTP protocol
Wirawan & Nugroho [11] (2023)	BLE only	Local only	±0,5°C	None	BLE sensor node, no cloud integration
Wardhana et al. [19] (2021)	MQTT + Wi-Fi	~450	±0,5°C	None	MQTT temp monitoring, no BLE bridging
This Work	BLE + MQTT	318 (mean)	±0,22°C RMSE	Plain MQTT	BLE–MQTT bridge, quantified latency, RMSE validation

REFERENCES

[1] V. A. Orfanos, S. D. Kaminaris, P. Papageorgas, D. Piromalis, and D. Kandris, "A Comprehensive Review of IoT Networking Technologies for Smart Home Automation Applications," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, p. 30, Apr. 2023, doi: 10.3390/jsan12020030.

[2] A. H. Wirasapta, T. D. Pamungkas, and S. S. Mishi, "Simulation-Based Analysis of Packet Scheduling Strategies and Traffic Load Effects on Network QoS," *Energy Insights*, vol. 1, no. 2, pp. 49–58, Apr. 2026, doi: 10.59562/ei.v1i2.11311.

[3] D. Arifianto, A. Sulistyono, and A. Nilogiri, "Sistem Monitoring Suhu Dan Kelembaban Ruangan Server Berbasis Arduino Menggunakan Metode Fuzzy Logic Dengan Buzzer Dan Telegram Bot Sebagai Notifikasi," *JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia)*, vol. 7, no. 1, pp. 67–75, Mar. 2022, doi: 10.32528/justindo.v7i1.5135.

[4] U. Ristian, I. Ruslianto, and K. Sari, "Sistem Monitoring Smart Greenhouse pada Lahan Terbatas Berbasis Internet of Things (IoT)," *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, vol. 8, no. 1, p. 87, Apr. 2022, doi: 10.26418/jp.v8i1.52770.

[5] Md. M. Islam, S. Nooruddin, F. Karray, and G. Muhammad, "Internet of Things: Device Capabilities, Architectures, Protocols, and Smart Applications in Healthcare Domain," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3611–3641, Feb. 2023, doi: 10.1109/JIOT.2022.3228795.

[6] I. Petrescu, E. Niculae, V. Vulturescu, A. Dimitrescu, and L. M. Ungureanu, "Transport and Application Layer Protocols for IoT: Comprehensive Review," *Technologies (Basel).*, vol. 13, no. 12, p. 583, Dec. 2025, doi: 10.3390/technologies13120583.

[7] P. Fauzan Prasetyo Eka Putra, M. Amir Mahmud, and R. Paradina, "Comparing the Performance of LoRaWAN and MQTT Protocols for IoT Sensor Networks," *Journal of Information and Technology*, vol. 6, no. 2, 2024, doi: 10.60083/jidt.v6i2.565.

[8] M. F. Zulkarnaen, Aliy Nauval Hanafi, and Mohammad Taufan Asri Zaen, "Rekayasa SmartHome System Berbasis Internet of Things," *Infotek: Jurnal Informatika dan Teknologi*, vol. 7, no. 2, pp. 552–562, Jul. 2024, doi: 10.29408/jit.v7i2.26545.

[9] A. A. Al Sarfini and D. Irawan, "Sistem Kontrol Jarak Jauh Plc Menggunakan Esp32 Berbasis Iot," *Jurnal Amplifier: Jurnal Ilmiah Bidang Teknik Elektro dan Komputer*, vol. 14, no. 1, pp. 51–55, May 2024, doi: 10.33369/jamplifier.v14i1.33484.

[10] J. O. Christopher, M. Resquites, M. A. Parrocho, N. Vinegas, D. R. Vinyl, and H. Oquiño, "IoT-Based Temperature Monitoring and Automatic Fan Control Using ESP32," *IRE Journals*, vol. 7, no. 5, pp. 35–44, 2023.

[11] A. P. Wirawan and H. Nugroho, "Perancangan Node Sensor Nirkabel BLE Bertenaga Baterai menggunakan ESP32 untuk Aplikasi Pertanian Cerdas," *Telekontran: Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 11, no. 1, pp. 12–22, May 2023, doi: 10.34010/telekontran.v11i1.9607.

[12] K. F. Sumartono and Y. T. Samuel, "Implementation of Internet of Things (IoT) System at PT XXX using the MQTT system," *Jurnal TeIKa*, vol. 14, no. 2, pp. 173–184, 2024.

[13] A. Sabo, H. O. Suleiman, Y. Dahiru, N. D. Jatau, A. Yusuf, and A. T. Chikodi, "Development and Implementation of an ESP32 IOT-Based Smart Grid for Enhanced Energy Efficiency and Management," *European Journal of Theoretical and Applied Sciences*, vol. 2, no. 3, pp. 565–576, May 2024, doi: 10.59324/ejtas.2024.2(3).43.

[14] MicroPython Team, "MicroPython Documentation." [Online]. Available: https://micropython.org/

[15] Dodi Yudo Setyawan, Nurfiana, Lia Rosmalia, Melia Gripin Setiawati, and Retno Dwi Handayani, "Kalibrasi Sensor Suhu Udara, Kelembaban dan pH Tanah Menggunakan Metode Linear regression," *Jurnal TEKNIKA*, 2024.

[16] Y. Im and M. Lim, "E-MQTT: End-to-End Synchronous and Asynchronous Communication Mechanisms in MQTT Protocol," *Applied Sciences*, vol. 13, no. 22, p. 12419, Nov. 2023, doi: 10.3390/app132212419.

[17] S. Yuan, "Review of Root-Mean-Square Error Calculation Methods for Large Deployable Mesh Reflectors," *International Journal of Aerospace Engineering*, vol. 2022, pp. 1–18, Aug. 2022, doi: 10.1155/2022/5352146.

[18] C. D'Ortona, D. Tarchi, and C. Raffaelli, "Open-Source MQTT-Based End-to-End IoT System for Smart City Scenarios," *Future Internet*, vol. 14, no. 2, p. 57, Feb. 2022, doi: 10.3390/fi14020057.

[19] I. Wardhana *et al.*, "Rancang Bangun Alat Pengukur Suhu Real Time Laboratorium Menggunakan Protokol MQTT Berbasis Internet of Things," *Jurnal Teori dan Aplikasi Fisika*, vol. 09, no. 01, pp. 39–46, 2021.

[20] F. Dewanta, B. Y. Yustiarini, and B. Indrakusumo Radityo Harsritanto, "A study of secure communication scheme in MQTT: TLS vs AES cryptography," *Jurnal INFOTEL (Informatics, Telecommunication, and Electronics)*, vol. 14, no. 4, pp. 269–276, Nov. 2022, doi: 10.20895/infotel.v14i4.807.