Proceedings in Manufacturing Systems, Volume 19, Issue 2, 2024, 61-68 ISSN 2067-9238

REAL-TIME DATA ACQUISITION WITH ESP32 FOR IOT APPLICATIONS USING OPEN-SOURCE MQTT BROKERS

Adrian MAROȘAN[1,*], George CONSTANTIN[2], Claudia Emilia GÎRJOB[3], Anca Lucia CHICEA[4], Mihai CRENGANIȘ[5], Fineas MORARIU[6]

[1)] Lecturer, PhD, Machines and Industrial Equipment, University "Lucian Blaga" of Sibiu, Faculty of Engineering, Romania
[2)] Prof., PhD, Robots and Manufacturing Systems Department, POLITEHNICA Bucharest, Romania
[3)] Assoc. Prof., PhD, Machines and Industrial Equipment, University "Lucian Blaga" of Sibiu, Faculty of Engineering, Romania
[4)] Assoc. Prof., PhD, Machines and Industrial Equipment, University "Lucian Blaga" of Sibiu, Faculty of Engineering, Romania
[5)] Assoc. Prof., Conf., Machines and Industrial Equipment, University "Lucian Blaga" of Sibiu, Faculty of Engineering, Romania
[6)] Assist. Prof., PhD Student, Machines and Industrial Equipment, University "Lucian Blaga" of Sibiu, Faculty of Engineering, Romania

Abstract: *This study introduces a novel approach to real-time data acquisition and monitoring using the versatile ESP32 microcontroller, which is highly regarded for its flexibility and capabilities in IoT solutions. Sensor data is transmitted wirelessly via an open-source MQTT broker, which has been specifically adapted to meet the scalability requirements of modern IoT infrastructures. This method not only facilitates real-time data capture but also provides a robust platform for processing and visualizing the collected information by integrating efficient software solutions. The paper delves into both the hardware configuration, such as setting up the ESP32 and connecting it to different sensors, as well as the software development required to manage communication protocols and implement MQTT integration. The MQTT protocol is particularly beneficial due to its efficiency in data transmission with minimal bandwidth usage and low latency, making it suitable for fast, reliable communications, which are critical in IoT environments. Experimental results show that the proposed solution ensures high stability and fast response times, making it applicable in various domains. These include the remote monitoring and control of smart homes, where precision and quick response are essential, automation of industrial processes, and the development of intelligent environments. Moreover, the solution demonstrates significant potential for integration into Industry 4.0, with the ability to work alongside emerging technologies such as augmented reality (AR) for system monitoring and maintenance in industrial production settings. As a result, this system can play a pivotal role in optimizing industrial processes by offering a comprehensive solution for real-time control and data analysis in crucial industrial applications.*

Key words: *Real-time data acquisition, ESP32, IoT, MQTT, Wireless communication, Industry 4.0, Sensor integration, Augmented reality (AR).*

1. INTRODUCTION

The Internet of Things (IoT) technology has revolutionized numerous industries, from home automation and healthcare to energy resource management, industrial robotics, and smart agriculture. Integrating IoT devices and using efficient communication protocols like MQTT has enabled real-time data collection and analysis, facilitating informed decision-making and process optimization. The analysed documents highlight the impact of IoT in these fields and present various innovative methods and solutions. In the context of home automation, IoT has become a key component for enhancing comfort and energy efficiency in modern homes. The proposed solutions use microcontrollers like ESP32, allowing the automation of electrical devices through mobile applications or web interfaces, using the MQTT protocol for fast and efficient communication. For example, the study on home automation based on ESP32 and Node-RED demonstrates how devices can be monitored and controlled remotely, improving safety and reducing energy costs [1]. This system leverages an economical and scalable infrastructure that allows users to easily integrate new smart devices into their homes. Healthcare is another domain where IoT has brought significant advancements, particularly in real-time patient monitoring. By using wearable sensors connected to ESP32 microcontrollers, patients' vital parameters, such as body temperature, blood oxygen levels, and heart rate, can be continuously monitored and transmitted to central platforms via MQTT. These solutions enable the early detection of health issues and automatic alerting in emergencies [2]. For instance, systems developed for smart homes dedicated to the elderly provide continuous monitoring and safety, enhancing their quality of life. In terms of energy management, IoT offers scalable and efficient solutions for optimizing resources and reducing

[*] Corresponding author: Emil Cioran nr.4, Sibiu,Romania, Tel.: +40742945750.
E-mail addresses: *adrian.marosan@ulbsibiu.ro* (A. Maroșan), *george.constantin@icmas.eu* (G. Constantin)

consumption. The use of IoT platforms allows detailed monitoring of energy usage through integrated sensors that send data to central servers. These platforms analyse consumption in real-time and provide recommendations for resource-saving measures [3]. A relevant example is the integration of IoT solutions in energy management systems, using platforms like Node-RED to visualize and analyse collected data, contributing to more efficient resource management and reduced environmental impact. Industrial automation has adopted IoT technologies to improve efficiency and precision in industrial processes. Industrial robots can be controlled and monitored in real-time using fast communication protocols like MQTT, reducing latency and optimizing commands. The study on monitoring an industrial robotic arm shows how MQTT enables precise and reliable control of robot movements, reducing costs and maintenance time [4] and [5]. These solutions are essential for modern industrial production, facilitating the transition to Industry 4.0 and improving factory safety. Smart agriculture is another field where IoT has made a significant impact, contributing to more efficient resource use and higher yields. Automated irrigation systems, based on humidity and temperature sensors, monitor environmental conditions in real-time and adjust water usage according to plant needs. The study on smart gardens proposes a system that uses ESP8266 to collect and transmit data about soil moisture and ambient temperature, optimizing water consumption and resources [6]. These solutions allow farmers to manage crops more efficiently, reducing costs and maximizing productivity. Moreover, environmental monitoring and air quality assessment have become critical concerns in the context of climate change and increasing pollution. IoT provides efficient solutions for collecting and analysing data on weather conditions and air quality, using integrated sensors with microcontrollers. One example is the use of smart weather stations that monitor temperature, humidity, and pollutant gas concentrations, providing essential information for environmental protection and public health [7]. These data are transmitted in real-time via MQTT to web platforms and mobile applications, offering users immediate access to environmental conditions. The analysed documents clearly illustrate the versatility of IoT technology and its capacity to bring significant improvements in various sectors, from industrial process automation to energy efficiency and resource optimization. The implementation of IoT in agriculture, healthcare, industry, and home automation not only contributes to a safer and more convenient life for users but also promotes a more sustainable and resource-friendly environment.

2. LITERATURE REVIEW

The study for this paper involved an extensive review of the current literature on IoT applications using ESP32 and open-source MQTT brokers. Table 1 includes a selection of 19 works, carefully chosen to cover a broad spectrum of IoT applications relevant to real-time data acquisition. These works span various domains, including smart home automation, healthcare monitoring, energy management, industrial robotics, and environmental sensing, showcasing the versatility and adaptability of the ESP32 microcontroller combined with MQTT protocols.

Table 1

Overview of Relevant IoT Studies Using ESP32 and MQTT Protocols for Real-Time Data Acquisition sensor.

Title	Year	Type of Work	Field
Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System	2023	Conference Paper	Energy, Internet of Things (IoT)
Open-Source MQTT-Based End-to-End IoT System for Smart City Scenarios	2022	Academic Article	Smart Cities, Internet of Things (IoT)
Implementation of Home Automation System Using MQTT Protocol and ESP32	2018	Conference Paper	Home Automation, Internet of Things (IoT)
Research on Real-time Monitoring System of Multiscenario Health Data Based on Internet of Things	2023	Conference Paper	Healthcare, Internet of Things (IoT)
Smart Saline Level Monitoring System Using ESP32 and MQTT-S	2018	Academic Article	Healthcare, Internet of Things (IoT)
The Use of the MQTT Protocol in Measurement, Monitoring, and Control Systems as Part of the Implementation of Energy Management Systems	2023	Academic Article	Energy Management , Internet of Things (IoT)

The research presented in these papers explores various applications of the ESP32 microcontroller and the MQTT protocol in IoT systems for real-time data acquisition, demonstrating the flexibility and efficiency of these technologies in multiple fields. The first paper analyzes the energy consumption of the ESP32 microcontroller in a real-time IoT monitoring system using the MQTT protocol, aiming to optimize energy usage without affecting the system's performance, which is a crucial feature for IoT applications that use battery-powered devices [8]. In a similar direction, another paper presents a complete IoT system based on open-source MQTT for smart city applications, highlighting the use of ESP32 for real-time data transmission from various applications such as environmental monitoring and urban

resource management. This open system allows scalability and efficient integration of IoT devices, making it suitable for large-scale deployment in urban environments [9]. The connection between these two studies lies in the use of ESP32 and MQTT for real-time communication, but with different purposes: the first focuses on energy efficiency, while the second focuses on integrating technologies into urban infrastructure.

In the same direction of IoT applications, another paper describes the implementation of a home automation system that uses ESP32 and MQTT for remote control and monitoring of household appliances. It emphasizes the use of real-time data to improve comfort and efficiency in homes, allowing users to interact with their devices from any location via a mobile application or web interface. This automation system highlights ESP32's ability to respond quickly to commands and transmit data efficiently [10]. The connection with the previous study is clear, as both research works address the use of ESP32 and MQTT for real-time resource management, this time in the context of a smart home. Expanding the use of ESP32 and MQTT into healthcare, one study investigates real-time health data monitoring using wearable sensors connected to ESP32, transmitting parameters such as heart rate and body temperature via MQTT to a centralized platform, enabling remote patient monitoring. This approach has significant applications in telemedicine, where continuous monitoring systems are required for patients [5].

Additionally, another study explores the use of ESP32 and the MQTT-S protocol, an optimized version of MQTT, in a saline level monitoring system, demonstrating the effectiveness of this system in transmitting real-time data with energy-saving conditions, which is essential for medical devices requiring low energy consumption [11].

Both studies highlight the applicability of ESP32 in healthcare, whether for continuous patient monitoring or for medical systems that must function long-term with low energy consumption. Finally, the last paper discusses the use of the MQTT protocol in energy management systems, where ESP32 is used to measure and monitor energy consumption in real-time, integrating IoT into energy control processes. This is an important step in optimizing energy usage in various industries and smart buildings, contributing to increased efficiency and reduced resource consumption [3].

This research connects with the other studies using ESP32 and MQTT to facilitate the efficient management of resources but applied in the energy sector. Therefore, these papers emphasize the importance and versatility of ESP32 and MQTT in developing efficient real-time IoT solutions, applicable across a wide range of fields, from home automation and healthcare monitoring to resource management in smart cities and optimizing energy consumption.

3. MATERIALS AND METHODS

In this paper, the goal is to develop a complete IoT system for monitoring water levels, using the ESP32 microcontroller and the MQTT protocol. The objectives of this research were defined based on the analysis of the specialized literature, which highlighted the requirements and challenges encountered in implementing real-time data acquisition IoT systems. Considering the importance of performance and efficiency in these applications, the aim of the research is to create a functional and scalable solution that meets the current needs of users. The main objectives of the research are outlined below, focusing on both hardware development and software implementation.

3.1. Evaluating the Performance of ESP32 in IoT Applications for Real-Time Data Acquisition

Develop a water level sensor integrated with ESP32 and create an Android application for real-time data visualization, demonstrating the ability of ESP32 to collect and transmit data in a functional IoT system.

Figure 1 presents a simplified hardware configuration for a water level monitoring system using the ESP32-WROOM-32D microcontroller. The diagram shows how the main components of the system are connected: an RGB LED, a water level sensor, and a buzzer. The ESP32-WROOM-32D is the main microcontroller of the system, responsible for reading data from the sensor and activating the RGB LED and the buzzer based on the water level. The key features of the ESP32 microcontroller are presented in Table 2. The RGB LED is connected to the ESP32 via three signal pins (red, green, and blue), each corresponding to an individual color, allowing for a different visual signal to be displayed based on the water level. The water level sensor is connected to an analog pin of the ESP32, which will measure the water level and send the data to the microcontroller to activate or deactivate the RGB LED and buzzer. The buzzer is connected to a pin of the ESP32 and emits an audible signal to alert the user when the water level exceeds a critical threshold.

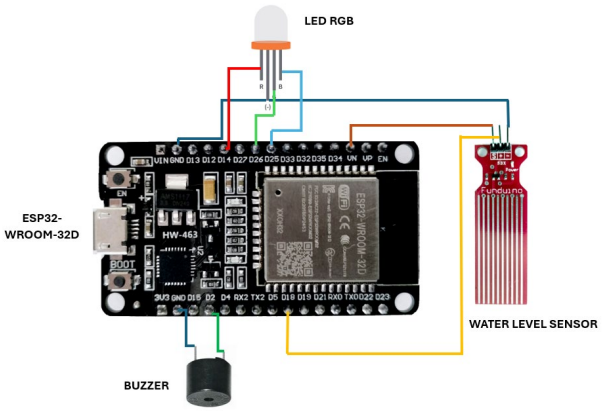

Fig. 1. Hardware Configuration for Water Level Monitoring System with ESP32.

Table 2

ESP32-WROOM-32D Specifications

Parameters	Description
Processor	Dual-core Tensilica LX6, up to 240 MHz
Memory	520 KB SRAM, 4 MB Flash
WiFi	802.11 b/g/n, 2.4 GHz
Bluetooth	BT 4.2 (Classic and BLE
GPIO	34 GPIOs, 18 ADC channels, 2 DACs
I/O Interfaces	SPI, I2C, UART, PWM, CAN

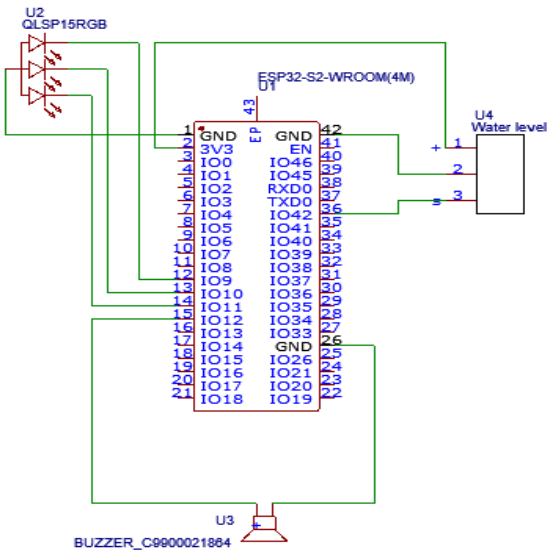

Fig. 2. Detailed Circuit Diagram for Water Level Monitoring System Using ESP32.

Figure 2 represents a detailed electrical schematic of the water level monitoring system, using the ESP32-S2-WROOM as the central microcontroller. The diagram outlines the specific connections of each component to the ESP32, providing a clear view of how the electronic components are interconnected. The ESP32-S2-WROOM is connected to various components via GPIO pins, with each pin labeled to indicate the corresponding component. These connections include both analog pins (for the water level sensor) and digital pins (for the RGB LED and buzzer). The RGB LED (labeled as U2) is connected to three GPIO pins, each controlling one of the LED's colors (red, green, and blue), signaling different water level states. The water level sensor (U4) is connected to an analog pin of the ESP32 to transmit the water level data to the microcontroller, while the buzzer (U3) is connected to another GPIO pin of the ESP32 to emit an audible signal when the water level reaches a critical threshold. The specific pins for each component are defined as follows: GPIO17 is connected to the sensor's VCC pin (POWER_PIN), GPIO36 (ADC0) is connected to the sensor's signal pin (SIGNAL_PIN), GPIO13 controls the red LED (for high water level) (RED_PIN), GPIO12 controls the green LED (for medium water level) (GREEN_PIN), GPIO14 controls the blue LED (for low water level) (BLUE_PIN), and GPIO15 is connected to the buzzer (BUZZER_PIN). This diagram is much more detailed, providing precise information about the pins, power connections, and the signals transmitted between components, making it useful for the physical realization of the circuit.

3.2. Implementing an Open-Source MQTT-Based IoT System for Real-Time Applications

In recent years, the integration of Internet of Things (IoT) technologies with real-time data acquisition systems has become a crucial aspect of modern technological solutions. Among the various communication protocols available, the MQTT protocol has gained significant popularity due to its lightweight, efficient, and open-source nature, making it ideal for real-time IoT applications. This objective of the paper focuses on implementing an open-source MQTT-based IoT system, using the ESP32 microcontroller as the central element. By utilizing MQTT, the system allows seamless data exchange between IoT devices, ensuring reliable, low-latency communication. This approach is particularly useful in environments where continuous monitoring and immediate action are necessary, such as home automation, healthcare, or environmental monitoring. The key advantage of using an open-source MQTT system lies in its flexibility, scalability, and cost-effectiveness, enabling easy adaptation and expansion of the system for a variety of real-time applications. In this context, the following sections will explore how the MQTT protocol facilitates real-time data transfer, ensuring that the system responds promptly to sensor readings and triggers appropriate actions, such as activating alerts or controlling devices.

The MQTT protocol offers enhanced security compared to the HTTP protocol due to its integration with cryptographic protocols. Unlike HTTP, which is more prone to security vulnerabilities, MQTT ensures secure communication by encrypting the transmitted data. Additionally, MQTT is a lightweight protocol with low latency, making it ideal for real-time data transmission in IoT applications. One of the key advantages of using MQTT, combined with cryptographic protocols, is that it ensures the integrity and confidentiality of the data. With MQTT, only relevant and necessary data is transmitted and stored, eliminating any ambiguity or unnecessary information. This not only enhances data security but also optimizes data storage, ensuring that sensitive information is well-protected. As a result, the combination of MQTT and cryptographic techniques helps maintain both the security and efficiency of the system, making it a preferred choice for secure and reliable communication in IoT applications [11]. There are studies that compare several popular open-source MQTT brokers, including Mosquitto and HiveMQ, in terms of their performance in edge computing scenarios. Mosquitto, known for its simplicity, exhibited the best response times, especially for small payloads (1KB - 10KB), making it ideal for resource-constrained devices. However, in more challenging network conditions, with large payloads (1MB), its performance varied but still maintained relatively stable latencies compared to other brokers. HiveMQ, being more complex and offering more features, had slightly higher latency than Mosquitto under optimal network conditions, and its performance became less consistent in unfavorable network conditions, with a significant increase in latency as message size grew. EMQX demonstrated better scalability in large-scale deployments and high-throughput scenarios, while RabbitMQ, though reliable, did not outperform Mosquitto or HiveMQ in terms of latency, especially in high jitter or packet loss conditions. The study also highlighted the impact of network factors such as latency, jitter, and packet loss on performance, with differences being more pronounced for larger message sizes. Thus, Mosquitto provided the best response times for small payloads, while HiveMQ and EMQX excelled in large implementations and complex

IoT applications. The choice of broker depends on the specific requirements of the application, such as hardware, message size, and network conditions [12]. HiveMQ is an advanced and scalable MQTT broker, ideal for IoT applications that require managing a large number of devices connected simultaneously and efficient real-time data transfer. It is particularly suitable for large-scale projects, such as water level monitoring, due to its ability to support millions of connected devices and its stable performance in complex network scenarios. HiveMQ is built to support both MQTT 3.1.1 and MQTT 5, providing high flexibility and scalability in managing data flows and messages. Support for MQTT 5 adds advanced features such as improved error handling and topic management, which helps manage communication between devices more efficiently. Another significant advantage of HiveMQ is its advanced security; the broker uses TLS encryption to ensure the confidentiality and integrity of data transmitted between IoT devices, thus protecting sensitive information. Additionally, HiveMQ offers robust authentication and authorization options, including support for certificate-based and token-based authentication, ensuring strict control over access to system data. Besides these features, HiveMQ is known for its reliability, ensuring message delivery even in challenging network conditions. In the case of connection interruptions, it offers automatic reconnection and message retention until full delivery. Furthermore, HiveMQ allows integration with various external systems and extension of functionalities through plugins, making IoT solutions based on HiveMQ highly flexible and customizable. These characteristics make HiveMQ the perfect solution for implementing an IoT-based water level monitoring system, which requires both fast data transfer and security [13].

Figure 3 illustrates the main services and areas where HiveMQ can be integrated, highlighting its diverse applications, ranging from industrial automation and energy management to smart cities and healthcare monitoring. The diagram shows HiveMQ's versatility in connecting a wide range of IoT devices from various industries, such as automotive, manufacturing, agriculture, healthcare, environmental monitoring, and home automation. HiveMQ enables efficient, secure, and real-time communication between IoT devices and backend systems in these fields. For example, in the automotive industry, HiveMQ facilitates the integration of sensors and devices that monitor vehicle status and enable communication between them and the transportation infrastructure. In manufacturing, HiveMQ supports communication between machines and production equipment, while in agriculture, HiveMQ can be used to connect sensors monitoring environmental conditions and crops.

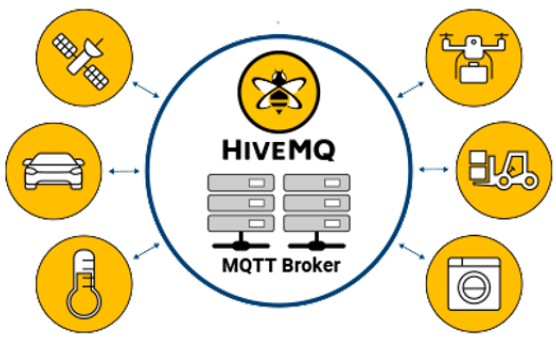

Fig. 3. HiveMQ Integration in Various IoT Applications [13].

In healthcare, HiveMQ is used for real-time data transfer from medical devices to central monitoring systems, while in environmental monitoring, the broker can manage data from temperature, humidity, or pollution sensors. In home automation, HiveMQ allows for the integration and control of smart devices, such as appliances and thermostats, enabling users to monitor and control them remotely.

This diagram emphasizes the importance of HiveMQ in supporting a wide range of IoT applications, demonstrating its ability to support not only simple applications but also complex and distributed solutions that require managing a large volume of data and devices. HiveMQ is an excellent choice for applications that require high performance, security, and scalability, and its flexibility makes it suitable for deployments of any size, from individual solutions to global systems connecting millions of devices.

HiveMQ was chosen as the solution for the water level monitoring application due to its scalability, performance, security, and reliability features, which are essential for the success of an IoT system of this type. My application involves real-time water level monitoring through sensors connected to the ESP32 microcontroller, and HiveMQ stands out due to its ability to support a large number of devices connected simultaneously, which is crucial as the system expands. Given that the application may include an increasing number of sensors distributed across multiple locations, HiveMQ ensures excellent scalability, allowing it to manage a high volume of data and connections without compromising performance. Additionally, data protection is a priority in my application, considering the sensitive nature of the transmitted information, and HiveMQ provides advanced security through TLS encryption, ensuring the confidentiality and integrity of all messages transmitted between IoT devices and the central server. HiveMQ also has authentication and authorization features, which allow for strict access control and protect data from unauthorized access. Reliability is another essential aspect, as HiveMQ guarantees message delivery even in challenging network conditions, and in case of connection interruptions, the broker can store messages and deliver them later, without losing critical information. These features are crucial for my system, which must operate in a dynamic environment where connection interruptions are inevitable but must not affect the system's reliability. HiveMQ's flexibility in integrating with various external platforms and its ability to extend functionalities through plugins make it easy to integrate with other monitoring systems, databases, or cloud platforms, which is important for the future development of the application. Support for MQTT 5 adds additional features, such as improvements in error handling and topic management, which facilitate better data flow management and greater flexibility in communication control. These improvements are essential for IoT applications that require efficient and

secure data transfer. Given all these advantages, HiveMQ is the optimal solution for my water level monitoring application, ensuring performance, security, reliability, and scalability, aspects that are fundamental for the long-term success of the implemented IoT system [13].

4. RESULTS

4.1. Testing and Calibrating the Water Level Sensor for Data Accuracy and Real-Time Performance

The testing and calibration phase is crucial for ensuring that the water level sensor in my application provides reliable and accurate readings under various conditions. Accurate data is essential for the proper functioning of the monitoring system, especially when the system needs to trigger alerts or actions based on the water level. This process involves validating the sensor's response, calibrating it to ensure precise measurements, and testing its real-time performance to guarantee it meets the operational requirements of my water level monitoring system. Proper calibration ensures the system provides accurate water level measurements, minimizing errors and enhancing the overall reliability of the application.

Figure 4 presents a complete system for water level monitoring using the ESP32 microcontroller, which integrates both hardware and software components for managing and transmitting data in an IoT (Internet of Things) network. The process begins with connecting the ESP32 to the WiFi network and an MQTT broker, which ensures the transmission of data from the water sensor to an application or server. MQTT, a lightweight and efficient messaging protocol for IoT, is essential for fast and secure communication, and the system constantly checks if the connection to the MQTT broker is established. If the MQTT connection is not successful, the system will attempt to reconnect repeatedly until the connection is active, ensuring continuous and reliable communication. Once the connection to MQTT is established, the ESP32 reads the data from the water sensor, processing it to calculate the water level using a method that converts the sensor signals into percentages. This calculation allows for an accurate evaluation of the water level, and depending on the result, the system activates different visual and audio signals to inform the user about the water status. If the water level is below 25%, the system signals with a blue LED, indicating that the water level is too low, which may suggest a supply issue or a drought risk. If the water level is between 25% and 80%, a green LED is lit, signaling a normal and safe water level. In contrast, if the water level reaches 80% or more, indicating a risk of flooding or other water management issues, the system turns on a red LED and activates a buzzer to alert the user audibly. Additionally, an alarm message is sent to the MQTT topic, notifying about the emergency and allowing quick intervention by operators or external control systems. This integrated system provides an efficient and reliable solution for real-time water level monitoring, combining the ESP32 hardware with MQTT software to create a robust IoT platform. In this way, the system not only ensures local visualization of the water status but also allows remote monitoring, which is essential in automation and control

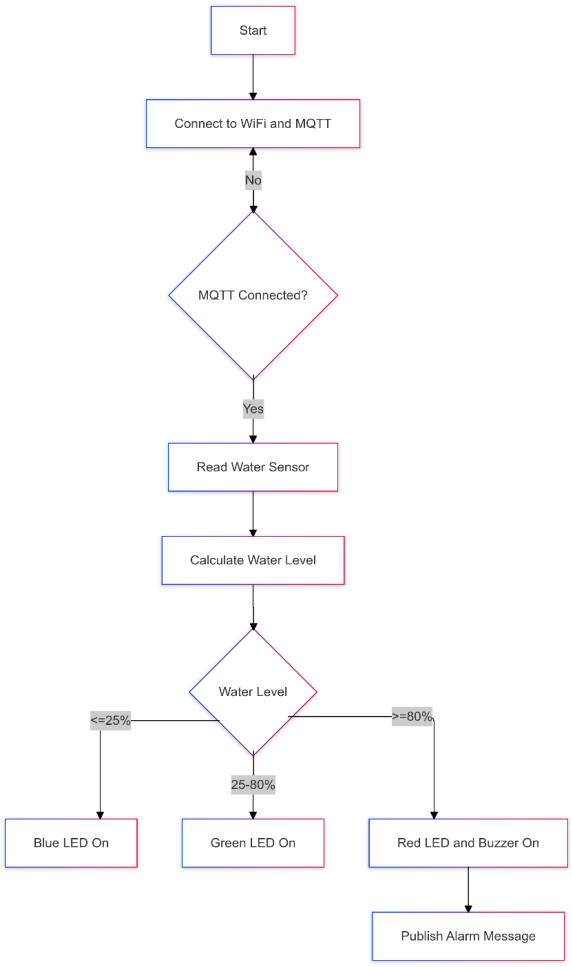

Fig. 4. ESP32 Water Monitoring System.

applications, such as water management in smart homes, industry, or agricultural environments. The system can be extended and adapted for use in various fields, such as industrial process automation or protection and monitoring solutions for critical infrastructures, demonstrating significant potential for integration into broader solutions like Industry 4.0, which integrates emerging technologies such as augmented reality for maintenance and monitoring in production.

To evaluate the functionality and performance of the real-time water level monitoring system based on the ESP32 microcontroller and MQTT protocol, an experimental system was implemented, as detailed in Figure 1. This system was tested in three distinct scenarios, each addressing different monitoring conditions, to analyze how the information is transmitted and received by the mobile application developed for smartphones. The tests were designed to verify the reliability of communication between the ESP32 microcontroller and the mobile application, as well as its ability to process and display the data efficiently and accurately. The results obtained from the three tested cases are presented below, offering a detailed view of the system's behavior under various operating conditions.

4.2. Real-Time Water Level Monitoring with ESP32 and MQTT: Case 1

Figure 5 presents a functional real-time water level monitoring system using the ESP32 microcontroller and

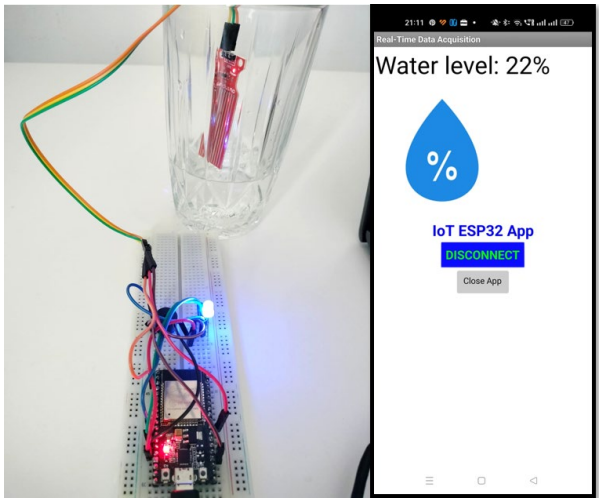

Fig. 5. Real-Time Water Level Monitoring with ESP32 and MQTT: Testing a Low Water Level.

MQTT protocol, as described in Case 1. In this experiment, the water level sensor, visible in the glass of water, is connected to the ESP32 microcontroller, and a blue LED lights up to indicate that the water level is below 25%, signaling a low water level. The measured values are transmitted quickly to the MQTT broker and displayed on the mobile application, showing a water level of 22% with a latency of about 100 ms.

The mobile application provides the user with full control over the system, including options to disconnect and close the app, offering an intuitive interface for continuous monitoring. This test demonstrates the efficiency and reliability of the monitoring system, providing accurate data and fast updates, which is essential for real-time IoT applications such as water resource management in households or industrial environments.

4.3. Real-Time Water Level Monitoring with ESP32 and MQTT: Case 2

Figure 6 illustrates the functioning of Case 2 in the water level monitoring system, where the water level detected by the sensor falls between 25% and 80%, which is considered a normal operating level. In this scenario, the system triggers the green LED, indicating that the water level is within a safe and acceptable range. At the same time, the real-time value of the water level, shown as 26%, is transmitted to the MQTT broker. This data is quickly received by the mobile application, displaying the water level with a latency of 100 ms. The app also provides the user with options to disconnect and close the application, ensuring that the system operates smoothly and provides real-time monitoring with minimal delay. This test successfully demonstrates how the system efficiently communicates and displays data, offering quick updates to users and ensuring that the water level remains in the designated normal range.

4.4. Real-Time Water Level Monitoring with ESP32 and MQTT: Case 3

Figure 7 illustrates the operation of Case 3 in the water level monitoring system. In this scenario, the water level detected by the sensor exceeds 80%, signaling a high water level. The system responds by activating the red LED to visually indicate the elevated water level. When the water level reaches 100%, indicating a potential overflow or flooding risk, the buzzer is triggered to emit a sound, providing an audible warning. In addition, the mobile application displays a pop-up notification alerting the user with the message "The water level is at 100%! Alarm!" This alert allows the user to take immediate action. The real-time water level, now at 100%, is sent to the MQTT broker and displayed on the mobile application with a latency of 100ms. This setup effectively demonstrates how the system provides both visual and audible alerts, ensuring that the user is immediately notified of critical water levels to prevent potential damage or flooding.

5. CONCLUSIONS

The ESP32 microcontroller has demonstrated its effectiveness in real-time water level monitoring, providing a 12-bit resolution, stable connectivity, and minimal latency. By leveraging an open-source MQTT broker, the system offers a cost-efficient and scalable solution, making it suitable for a variety of applications, from home monitoring to industrial processes. The mobile application allows users to easily track water levels and receive notifications directly on their

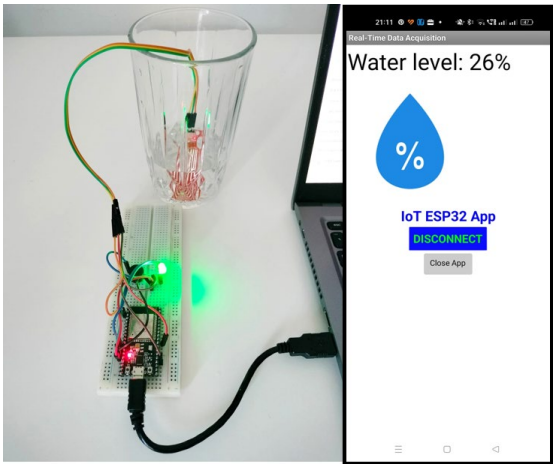

Fig. 6. Real-Time Water Level Monitoring with ESP32 and MQTT: Testing a Normal Water Level.

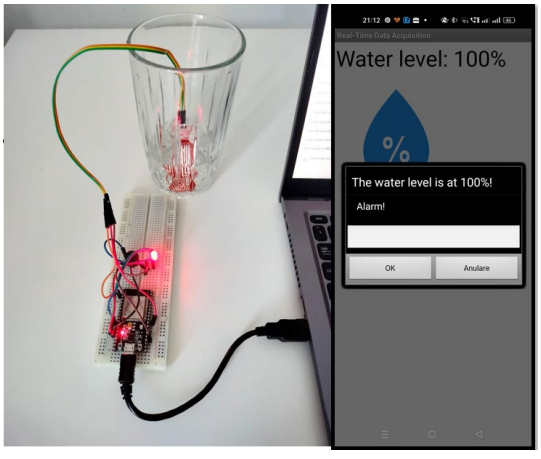

Fig. 7. Real-Time Water Level Monitoring with ESP32 and MQTT: Testing a High Water Level with Alarm.

smartphones, ensuring that they are promptly alerted to any critical changes in the water level. This feature improves the practicality of the system, especially in scenarios where rapid action is required, such as flood prevention, water resource management, or agricultural applications.

Looking to the future, several research directions could further improve the system. Integrating multiple sensors would enhance the system's ability to monitor various environmental factors, improving the accuracy and reliability of the data. In smart home applications, the system could be expanded to include automated water level control, allowing for real-time management of water usage and automatic flood detection. For industrial applications, the system could be scaled to monitor fluid levels in various processes, optimizing resource usage, reducing waste, and improving safety. Moreover, the system could be adapted for healthcare, where real-time monitoring of medical devices and patient safety would be critical. These potential developments demonstrate the versatility and broad applicability of the ESP32-based water level monitoring system across diverse sectors, paving the way for smarter, more efficient, and sustainable solutions in the future.

Funding: This research was funded by "Lucian Blaga" University of Sibiu (Knowledge Transfer Center) & Hasso Plattner Foundation research grants LBUS-HPI-ERG-2023, grant LBUS-HPI-ERG-2023-02.

REFERENCES

[1] A. K. Arigela, "Remote based Home Automation with MQTT : ESP32 Nodes and Node-RED on Raspberry Pi," *2024 8th Int. Conf. I-SMAC (IoT Soc. Mobile, Anal. Cloud)*, pp. 313–318, 2024, doi: 10.1109/I-SMAC61858.2024.10714655.

[2] S. Li, "IoT Healthcare System based on ESP32 for Smart Home," *2023 IEEE Int. Conf. Mechatronics Autom. ICMA 2023*, pp. 1768–1772, 2023, doi: 10.1109/ICMA57826.2023.10216003.

[3] A. Manowska, A. Wycisk, A. Nowrot, and J. Pielot, "The Use of the MQTT Protocol in Measurement, Monitoring and Control Systems as Part of the Implementation of Energy Management Systems," *Electron.*, vol. 12, no. 1, 2023, doi: 10.3390/electronics12010017.

[4] R. A. Atmoko and D. Yang, "Online Monitoring & Controlling Industrial Arm Robot Using MQTT Protocol," *Proc. 2018 Int. Conf. Robot. Biomimetics, Intell. Comput. Syst. Robionetics 2018*, pp. 12–16, 2019, doi: 10.1109/ROBIONETICS.2018.8674672.

[5] H. Wang, J. Huo, J. Hu, and T. Wang, "Research on Real-time Monitoring System of Multi-scenario Health Data Based on Internet of Things," *2023 3rd Int. Conf. Digit. Soc. Intell. Syst. DSInS 2023*, pp. 332–336, 2023, doi: 10.1109/DSInS60115.2023.10455560.

[6] D. Kurniawan, R. J. Putra, A. Bella, M. Ashar, and K. Dedes, "Smart Garden with IoT Based Real Time Communication using MQTT Protocol," *7th Int. Conf. Electr. Electron. Inf. Eng. Technol. Breakthr. Gt. New Life, ICEEIE 2021*, pp. 1–5, 2021, doi: 10.1109/ICEEIE52663.2021.9616869.

[7] M. Fahim, A. El Mhouti, T. Boudaa, and A. Jakimi, "Modeling and implementation of a low-cost IoT-smart weather monitoring station and air quality assessment based on fuzzy inference model and MQTT protocol," *Model. EARTH Syst. Environ.*, vol. 9, no. 4, pp. 4085–4102, Nov. 2023, doi: 10.1007/s40808-023-01701-w.

[8] M. A. Syahmi Md Dzahir and K. Seng Chia, "Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System," *2023 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2023*, pp. 255–261, 2023, doi: 10.1109/3ICT60104.2023.10391358.

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

[10] V. Thirupathi and K. Sagar, "Implementation of home automation system using mqtt protocol and esp32," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 2C2, pp. 111–113, 2018.

[11] D. Ghosh, A. Agrawal, N. Prakash, and P. Goyal, "Smart saline level monitoring system using ESP32 and MQTT-S," *2018 IEEE 20th Int. Conf. e-Health Networking, Appl. Serv. Heal. 2018*, pp. 1–5, 2018, doi: 10.1109/HealthCom.2018.8531172.

[12] J. Dizdarevic, M. Michalke, and A. Jukan, "Engineering and Experimentally Benchmarking Open Source MQTT Broker Implementations," 2023, [Online]. Available: http://arxiv.org/abs/2305.13893.

[13] `https://www.hivemq.com/`, accessed: 2024-11-29.