IoT Applications in Remote Patient Monitoring

Umut Akti
Engineering and Physical Sciences
Heriot Watt University
Edinburgh, UK
ua2018@hw.ac.uk

Mauro Dragone
Engineering and Physical Sciences
Heriot Watt University
Edinburgh, UK
m.dragone@hw.ac.uk

Abstract— **This report describes the design and implementation of an ESP32 microcontroller, DHT11 sensor, MQTT protocol using Mosquitto, and Node-RED dashboard for a real-time Internet of Things-based environmental monitoring system. Temperature and humidity data are efficiently collected and processed by the system, which displays the results in real time on an intuitive dashboard. Low-latency data transfer is made possible by the ESP32 microcontroller and the MQTT protocol, while Node-RED makes data management and presentation simple. The system's performance strengths and connection limitations brought on by campus network constraints were discovered through testing under various network settings. Despite issues with local network configuration and signal dependability, the results highlight the setup's feasibility for resource-efficient monitoring applications. Future developments could improve the system's accuracy, scalability, and adaptability across applications in environmental management, smart buildings, and agriculture. These developments could include sensor upgrades, cloud-hosted MQTT services, and machine learning integration.**

Keywords—IoT, Node RED, MongoDB, active sensors, cloud platforms, environment

I. Introduction

The internet has fundamentally transformed communication, enabling innovations that reach across industries, including healthcare, industrial automation, and environmental monitoring [1]. Over time, the demand for real-time data processing and analysis has intensified, leading to the development of specialized networked systems known as the Internet of Things (IoT) [2]. IoT connects physical devices such as sensors, actuators, and microcontrollers, allowing them to communicate over the internet to gather, transmit, and process data in real time. This capability is essential for applications where rapid insights and precise control are critical, such as environmental monitoring [3].

For environmental monitoring systems, lightweight and efficient communication protocols are paramount. MQTT (Message Queuing Telemetry Transport) has emerged as a preferred protocol for IoT, designed for low-bandwidth, resource-constrained environments [4]. The Mosquitto MQTT broker, an open-source MQTT implementation, plays a central role in this study, enabling efficient communication between devices in the IoT network [5].

The ESP32 microcontroller serves as the primary device in this system, providing low-power connectivity and multiple GPIO pins, which make it suitable for interfacing with various sensors and actuators, including the DHT11 temperature and humidity sensor [6]. The DHT11 sensor is a low-cost, reliable choice for basic environmental monitoring tasks, offering temperature and humidity readings at an accuracy level sufficient for typical IoT applications [7].

The software tools for this project include the Arduino Integrated Development Environment (IDE), which enables seamless programming and configuration of the ESP32 through custom sketches [8]. The Arduino IDE’s compatibility with multiple operating systems and its support for various microcontrollers make it a widely-used tool in prototyping IoT systems [9].

Another core software component is Node-RED, a visual programming environment built on Node.js and specifically designed to streamline IoT development by facilitating connections between sensors, data processing services, and cloud platforms. Created by IBM, Node-RED allows users to create data flows through a browser-based interface, making it an ideal choice for prototyping IoT applications and developing intuitive dashboards [10]. Node-RED’s dashboard functionality allows for real-time visualization of environmental sensor data, enhancing user interaction and system monitoring.

Together, the ESP32, DHT11, Mosquitto MQTT, and Node-RED dashboard form a robust IoT framework for real-time environmental monitoring with minimal latency. Figure 2. System Architecture Diagram illustrates data flow from the DHT11 sensor through the ESP32, Mosquitto broker, and Node-RED. This paper examines the system’s performance and compares findings with existing IoT-based monitoring solutions.

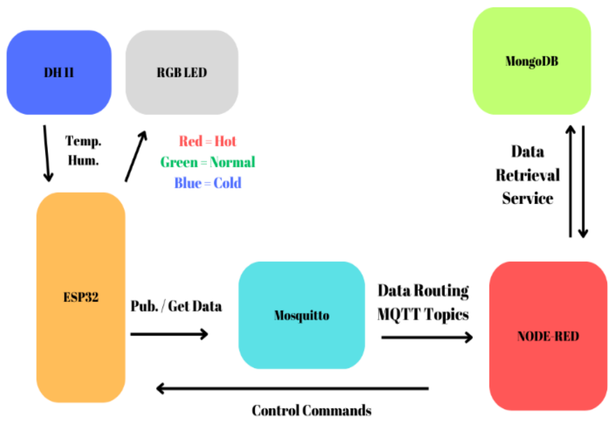

Fig. 1. General overview of the system

II. SYSTEM DEFINITION

A. *System Overview*

The primary goal of this IoT system is to monitor environmental conditions in real time, capturing temperature and humidity data from the surrounding environment. This data is collected by an ESP32 microcontroller, which interfaces with a DHT11 sensor and sends the data to a locally hosted Mosquitto MQTT broker. The MQTT broker manages message transmission and relays the data to a Node-RED dashboard, where it can be visualized and analyzed.

The ESP32 was selected for its dual-core processing power, built-in WiFi, and GPIO flexibility, making it suitable for low-power, continuous data transmission in IoT applications [6]. As the system's primary communication protocol, MQTT facilitates efficient, low-latency messaging, which is ideal for resource-constrained devices operating in real-time scenarios [4]. Node-RED provides the system's interface, presenting a dashboard where users can view current temperature and humidity levels and monitor historical trends, offering a flexible and accessible interface for non-technical users [10].

B. *Data Flow:*

The data flow within this system follows a sequence from data acquisition to visualization, ensuring efficient data handling and minimal latency. The process is as follows:

Data Acquisition: The DHT11 sensor reads temperature and humidity levels and sends this data to the ESP32 microcontroller. The DHT11 sensor is known for its reliability and cost-effectiveness, making it a popular choice for basic IoT monitoring applications [3]. (Insert image: "DHT11 Sensor Setup with ESP32")

Data Processing and Transmission: Once the ESP32 receives data from the DHT11 sensor, it publishes the data as MQTT messages. The locally hosted Mosquitto MQTT broker, running on a PC, acts as an intermediary, managing message distribution between the ESP32 and the Node-RED dashboard. MQTT's lightweight nature ensures efficient message handling, crucial for maintaining real-time updates in IoT environments [5]. (Insert image: "ESP32 with MQTT Broker Connection")

Data Visualization: Node-RED subscribes to the MQTT messages from the Mosquitto broker, processes the incoming data, and displays it in real-time through a dashboard. The dashboard includes interactive gauges and charts for temperature and humidity, allowing users to easily interpret environmental conditions [10]. Node-RED's intuitive, flow-based interface simplifies the development and modification of data visualization elements.

To provide a clear understanding of the hardware utilized, Figure 1 illustrates the main components in the system, including the ESP32 microcontroller, DHT11 sensor, anode LED, and circuit heater. This visual overview offers a concise reference for each part's physical design and setup.

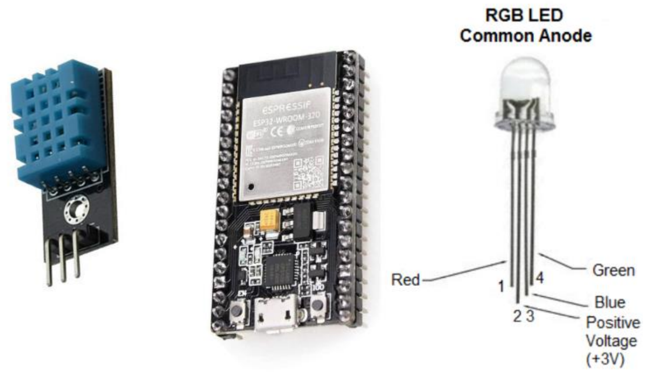

Fig. 2. Components used in the system.

III. METHODOLOGY

A. *Hardware Setup*

The hardware setup begins with connecting the DHT11 sensor to the ESP32 microcontroller to measure temperature and humidity, using the GPIO pins for communication. The anode LED is also connected to a GPIO pin to indicate data transmission status.

To simulate changes in environmental conditions and test the system's response, a circuit heater was positioned near the DHT11 sensor. This setup allowed us to observe the sensor's accuracy and the real-time data transmission under varying temperatures, validating the system's effectiveness for environmental monitoring.

B. *Software Configuration*

Arduino IDE Configuration: The Arduino IDE was used to program the ESP32, allowing it to read data from the DHT11 sensor and transmit it via MQTT. Libraries for the DHT11 sensor and MQTT protocol were imported to facilitate these processes. The following steps summarize the configuration:

- Library Imports: The DHT and MQTT libraries were imported to read data and manage MQTT communication.
- Setup: ESP32 was connected to the WiFi network, and MQTT topics for temperature and humidity were defined.
- Loop: The ESP32 continuously reads data from the DHT11 sensor and publishes it to the MQTT broker.

MQTT Broker Setup: The Mosquitto MQTT broker was installed on a local PC to manage data transmission between the ESP32 and Node-RED. Configuration steps included:

- Broker Setup: After installation, Mosquitto was set to listen on port 1883, the default MQTT port.
- Topic Definition: MQTT topics were defined as "environment/temperature" and "environment/humidity," with the ESP32 as the publisher and Node-RED as the subscriber. This ensured organized data flow from the sensor to the dashboard.

Node-RED Dashboard Setup: The Node-RED dashboard was configured to subscribe to the MQTT topics and display the data in real time. Key steps include:

- *Flow Creation:* A flow was created to receive MQTT messages from the broker, process the data, and update gauges and charts for temperature and humidity.
- *Dashboard Design:* The dashboard was customized with visual elements, including gauges for instant readings and a line chart to track historical data trends. This provided an accessible interface for interpreting environmental data effectively.

C. *Improving Chronic Disease Management*

Regarding chronic ailments such as diabetes or hypertension, IoT devices provide continuous, real-time monitoring that aids in managing symptoms without the necessity for frequent medical visits. Kakria et al. (2015) demonstrated that IoT solutions for cardiac patients resulted in substantial enhancements by facilitating prompt interventions during critical incidents [9]. These solutions facilitate improved long-term illness management and less hospitalizations.

D. *Challenges and Limitations*

Although the IoT presents certain advantages, it also poses concerns, particularly with data privacy and security. Griggs et al. (2018) investigated blockchain for secure data transmission in healthcare IoT systems; nonetheless, issues about scalability and cost persist [1]. Moreover, usability challenges for elderly patients and the necessity for additional research to enhance these systems for broader demographics remain pressing issues [7].

IV. Discussion

A. *Performance Evaluation*

The system demonstrated reliable data acquisition and real-time monitoring capabilities under controlled conditions. The ESP32 microcontroller successfully interfaced with the DHT11 sensor and transmitted data through the MQTT protocol to the Node-RED dashboard. The use of the Mosquitto MQTT broker ensured low-latency, efficient data transmission, which is essential in IoT applications requiring real-time updates [6]. However, the setup was occasionally hindered by connectivity issues, particularly on campus, where the network signal strength was inconsistent. Due to these limitations, a cellular network was often used as a backup to ensure consistent data flow, but this occasionally led to latency and increased data usage costs, which are known constraints in IoT environments [4].

B. *Comparison with Existing Literature*

Several studies have highlighted the efficiency of MQTT for lightweight, low-bandwidth data transmission in IoT applications, particularly in resource-constrained devices like the ESP32 [5]. This finding aligns with our system's successful performance in low-latency data handling under stable network conditions. Other studies utilizing Node-RED, such as those by Lekic and Gardasevic, show its effectiveness in simplifying IoT setups and providing user-friendly dashboards for data visualization [10]. Our findings support these conclusions, demonstrating that Node-RED's flow-based interface is indeed accessible and highly customizable, enabling straightforward monitoring of environmental data.

However, network connectivity and signal power limitations, as encountered on campus, illustrate a critical challenge often cited in IoT literature. Weak WiFi signals or frequent network drops can disrupt data continuity, which is essential for real-time applications [3]. Such connectivity issues underscore the importance of reliable infrastructure for IoT implementations in settings where network access may fluctuate. Similar studies have also noted this constraint and recommended using alternative solutions, such as deploying a dedicated IoT gateway or expanding local network coverage to improve reliability [4].

C. *Limitations and Suggested Improvements*

In addition to network-related issues, setting up the Mosquitto MQTT broker on a local PC presented occasional IP conflicts, complicating communication between the ESP32 and Node-RED. These IP conflicts affected the system's stability and required manual reconfiguration, which would not be ideal for scalable IoT deployments. Other studies recommend hosting the MQTT broker on cloud servers or using dynamic IP configurations to mitigate such issues [8]. Future improvements to this system could include moving the MQTT broker to a cloud-based service or implementing a more stable network configuration with dynamic IP support.

Regarding the sensor component, the DHT11 was effective for basic environmental monitoring; however, its limited accuracy could be improved by upgrading to a DHT22 sensor, which offers greater precision for both temperature and humidity measurements [7]. As seen in other studies, this upgrade could enhance the reliability of environmental monitoring, especially in applications requiring high data fidelity [3].

D. *Potential Applications and Future Work*

This IoT-based environmental monitoring system shows promise for applications in vaious real-world scenarios, such as agriculture, indoor air quality management, and industrial monitoring. Similar studies have shown that using IoT for real-time data collection in agriculture can help optimize irrigation schedules and improve crop yield [4]. In future iterations, this system could be expanded to include additional sensors (e.g., soil moisture or air quality sensors), further broadening its applicability.

Additionally, exploring machine learning techniques to analyze trends in temperature and humidity data could provide predictive insights. For instance, patterns in environmental data could be used to forecast weather conditions or assess building energy efficiency, as supported by studies on IoT-based predictive maintenance in smart environments [6]. These advancements could extend the system's usefulness beyond baic monitoring, contributing to intelligent and automated decision-making systems.

V. Conclusion

This study illustrates the capabilities of an IoT-based system for real-time environmental monitoring, leveraging an ESP32 microcontroller, DHT11 sensor, MQTT protocol with a Mosquitto broker, and a Node-RED dashboard. The system successfully captured, processed, and visualized temperature and humidity data, demonstrating practical benefits for applications requiring low-latency data transmission and efficient, accessible interfaces. Although network and IP configuration challenges were encountered, the system proved reliable under stable conditions, underscoring the potential of MQTT and Node-RED for IoT implementations in resource-constrained settings.

Several enhancements could improve system performance and broaden its applicability. Upgrading the DHT11 sensor to a more accurate model, such as the DHT22, would enhance data fidelity, while addressing network stability through a dedicated IoT gateway or cloud-hosted MQTT broker could ensure reliable communication across varied environments. Expanding the system's sensor suite to include soil moisture or air quality sensors would increase its versatility in applications like agriculture or smart buildings. Additionally, integrating machine learning for predictive analytics could enable the system to identify trends or anomalies in environmental data, allowing for optimized decision-making and automated responses.

This study establishes a solid foundation for scalable, IoT-based environmental monitoring systems and highlights directions for future work that could enhance IoT's role in intelligent, adaptive solutions across a range of applications.

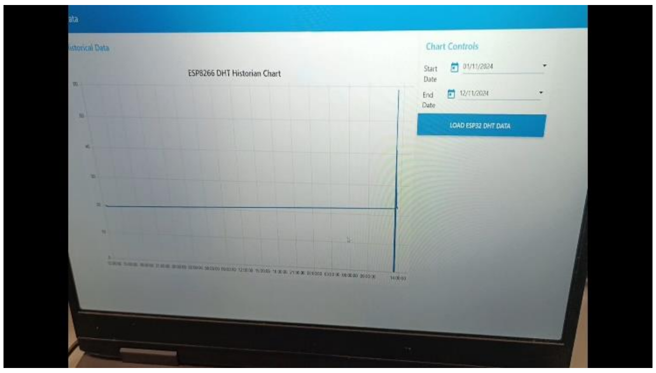

Fig. 3. Real time output graph of sensor data.

Appendix

Github link:

https://github.com/UmutAkti888?tab=repositories

References

[1] K. N. Griggs, O. Ossipova, and C. P. Kohlios, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," J. Med. Syst., vol. 42, no. 8, 2018.

[2] B. Pradhan and S. Bhattacharyya, "IoT - based applications in healthcare devices," J. Healthc. Eng., vol. 2021, 2021.

[3] A. Dixit, "IoT Applications with the DHT11 and DHT22 Sensors: An Introduction to Temperature and Humidity Monitoring," in IoT Fundamentals, 2nd ed., McGraw Hill, 2021.

[4] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," J. Open Source Softw., vol. 2, no. 13, pp. 265, 2017.

[5] Milica Lekic and G. Gardasevic, "IoT sensor integration to Node-RED platform," 17th Int. Symp. INFOTEH-JAHORINA, 2018.

[6] P. Verma and S. K. Sood, "Fog assisted-IoT enabled patient health monitoring in smart homes," IEEE Internet Things J., vol. 5, no. 3, pp. 1789–1796, 2018.

[7] M. Hassanalieragh, A. Page, and T. Soyata, "Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges," in Proc. IEEE Int. Conf. Serv. Comput., 2015, pp. 285–292.

[8] E. Upton and G. Halfacree, "The Official ESP32 Microcontroller Guide," Wiley, 2020.

[9] "Node-RED for IoT Development," IBM DeveloperWorks, 2020, [Online].

[10] R. P. Singh, M. Javaid, A. Haleem, and R. Suman, "Internet of things (IoT) applications to fight against COVID-19 pandemic," Diabetes Metab. Syndr. Clin. Res. Rev., vol. 14, no. 4, pp. 521–524, 2020.