

Design and Implementation of an End-to-End IoT System for Smart Healthcare

Shahad Ali Ridha ^{*}, Mohammed Issam Younis 

Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq

ABSTRACT

In recent years, there has been exponential growth in the Internet of Things (IoT) in various sectors, especially healthcare. The traditional healthcare sector has faced several challenges, such as poor health services, crowding and queuing at medical centers, and manual searches for or missing patient records; a smart health system is needed to overcome these challenges. This study presents the design and implementation of end-to-end IoT architecture, which is built by the integration of various open-source technologies. The pulse care system is implemented using Node.js and React.js, which offer features such as electronic health records (EHR), smart appointments, prescription documentation, and health monitoring. Additionally, various users (admin, patient, doctor, and pharmacist) communicate using a user-friendly graphical interface. In terms of a wearable device, it allows visualization of all the data generated by the sensors in real time (heart rate, body temperature, and blood oxygen). The ESP32 is used as a microcontroller that communicates via its built-in Wi-Fi with the MQTT Mosquitto broker. Standard IoT protocols, Message Queuing Telemetry Transport (MQTT) and Hypertext Transfer Protocol (HTTP), are used. The broker's performance is evaluated in terms of average latency at various QoS levels and with the increase in the number of concurrent users. The results of the experiment demonstrate how important it is to choose the appropriate QoS level based on the requirements of the application. Furthermore, the proposed system's data rate is roughly 3.07 kb/s, which is considered low and suitable for sending small data via any wireless method.

Keywords: IoT, MQTT, Electronic health records (EHR), Smart appointments, Remote health monitoring.

1. INTRODUCTION

The Internet of Things (IoT) represents an important technological revolution, and recently it has gained significant importance in human daily life. IoT connects a network of devices, such as cloud services, gateways, sensors, and actuators, to gather and exchange data.

*Corresponding author

Peer review under the responsibility of University of Baghdad.

<https://doi.org/10.31026/j.eng.2025.10.06>



This is an open access article under the CC BY 4 license (<http://creativecommons.org/licenses/by/4.0/>).

Article received: 25/04/2025

Article revised: 15/07/2025

Article accepted: 22/07/2025

Article published: 01/10/2025



(Younis et al., 2015; Kadhim and Hamad, 2023; Qadir and Hussan, 2023; Mahmood et al., 2023; Li et al., 2024). IoT applications have become increasingly important across a number of industries such as manufacturing, transportation, environmental monitoring, healthcare, and smart homes (Khan et al., 2023). The importance of IoT applications is particularly evident in the healthcare industry, which has improved diagnosis and treatment, and the ability to remotely monitor the fundamental signs of patients in real time, utilizing wearable devices. It can be used in many different medical areas, such as managing private health and fitness, supervising chronic illnesses, and caring for pediatric and elderly patients (Alattar and Mohsen, 2023). In addition, a smart healthcare system allows patients, particularly those suffering from chronic conditions, to live their lives more freely. Through the use of sensors, the patient's vital signs are continuously collected and sent for storage electronically. Physicians and patients will have access to this data through smart healthcare applications (Al-Fuqaha et al., 2015; Mohammed, 2024). Heart rate, blood pressure, blood oxygen saturation, respiration rate, and ECG are the most important vital signs (Elliott and Coventry, 2012). For tracking and monitoring health conditions, health monitoring systems are easy, intelligent, scalable, and effective (Islam et al., 2020).

Several key indicators influence health in the human body; the first indicator is heart rate, which measures the heart's beats or contractions per minute (Hegde et al., 2021). Blood oxygen level is the second indicator, and a lack of it can have numerous acute negative consequences, such as issues with the heart, kidneys, and brain. The third indicator is the body temperature, which is a measure of how hot or cold the body is (Islam et al., 2020). According to (Santoso et al., 2015), fever is essentially thought to be the typical sign of the majority of illnesses. The medical records are private documents that are maintained for each patient by medical professionals. They include the patient's personal information, tests, diagnoses, and treatments for any conditions the patient may be experiencing (Droma et al., 2009). Most clinics use patient records based on paper, which contain all the relevant medical information about a specific patient. This manual or paper-based record management approach is linked to several problems, such as being time-consuming, slow, inaccurate, and inefficient. In light of this, it becomes necessary to automate the current manual systems. With the rapid development of information and communication technology, the importance of efficient record-keeping in decision-making enhances human life and improving the quality of healthcare service. As a result, new tools for preserving patient data in digital format have been developed. One such tool to assist the medical facility is the Electronic Health Record (EHR). The EHR is intended to take the role of paper records as the main record-keeping format in the various healthcare facilities worldwide. In order to eliminate traditional paper-based medical records, (Joseph et al., 2020) developed an online platform for managing medical records. This automated system was developed using the following technologies: MySQL as the database engine, HTML, CSS, JavaScript, and PHP as scripting languages. However, the technologies used in this study are conventional and can be improved and have more features added.

In terms of communication, IoT provides several popular protocols, such as HTTP, SOAP, COAP, MQTT, and others. When compared to various protocols, (Turnip et al., 2020) found that MQTT was considered the most reliable and lightweight and was a power-saving technology. In order to improve healthcare services, some studies on the health monitoring system based on IoT have been implemented. Several systems with similar objectives and characteristics have lately been presented in this field, which has the primary goal of creating a smart environment that monitors people's health and provides the necessary



services (**Durán-Vega et al., 2019; Valsalan et al., 2020; Sangeethalakshmi et al., 2023**). A microcontroller and sensors form an Internet of Things-based health monitoring system. The study by (**Mohammed et al., 2023**) suggested a health monitoring system that uses Raspberry Pi 4B to gather patient data. It is a high-performance microcontroller but is considered large in size and not suitable for wearable devices. Various studies utilized the Arduino board as a microcontroller, including the ones suggested by (**Brashdi et al., 2018; Mostafa et al., 2020; Ruman et al., 2020; Kumar et al., 2022**). Numerous use cases (**Carducci et al., 2019; Babiuch et al., 2021**) for low-cost ESP32 microcontroller are documented in the literature; these applications range from Internet of Things solutions to more complex systems. The features of this microcontroller that differentiate it from other devices make it an appropriate choice for IoT devices. The wireless interface circuitry is the primary distinction between the ESP32 and Arduino Uno. The built-in Wi-Fi of the ESP32 makes it a remarkable microcontroller that has Wi-Fi capabilities. In this study, IoT architecture for the smart healthcare industry was implemented through combining different open-source technologies. The paper's primary contribution can be summed up as follows:

- Build a wireless smart end-to-end healthcare system based on IoT.
- Build a Pulse Care web application using Node.js and React.js, which provides online appointments, access to patient medical records, and real-time monitoring of data gathered by wearable devices.
- The patient's vital signs are monitored remotely and in real time by a wearable device.
- The MQTT communication protocol is used to transfer sensor data, and the HTTP protocol is used to transmit web application data.

2. THEORETICAL BACKGROUND

2.1 IoT Technology

The Internet of Things (IoT) refers to the connectivity between billions of physical devices around the world connected to the Internet through smart technologies, accessible from any location at any time. IoT connects multiple devices, such as sensors, actuators, and gateways that exchange data in networks that are included (**Balaji et al., 2019; Alshehri et al., 2021**). Devices that are linked to the internet and to one another can be used to assist people, remotely monitor and control systems, automate a number of tasks, and give real-time data that can be used to improve decision-making.

2.2 The Hardware Components

2.2.1 ESP32 Microcontroller

This study uses the ESP32 as the main microcontroller of the proposed wearable device. It represents a system on a chip (SoC); it contains a processor, memory, and communication units in one chip. Due to its small size, low cost, and low-power computation, it is suitable for IoT applications (**Espressif Systems, 2021**). ESP32 has a dual-core MCU with built-in Wi-Fi and Bluetooth. It includes a 32-bit LX6 CPU with a clock speed of 240 MHz, 448 KB of ROM, 520 KB of SRAM, and 4 MB of flash memory (Makeability Lab). This microcontroller has an I2C connection interface (Circuit Basics), which is used to connect sensors. In addition, they do not require the large computational power of a GPU to transmit sensor data. **Fig. 1** shows the ESP32 microcontroller.

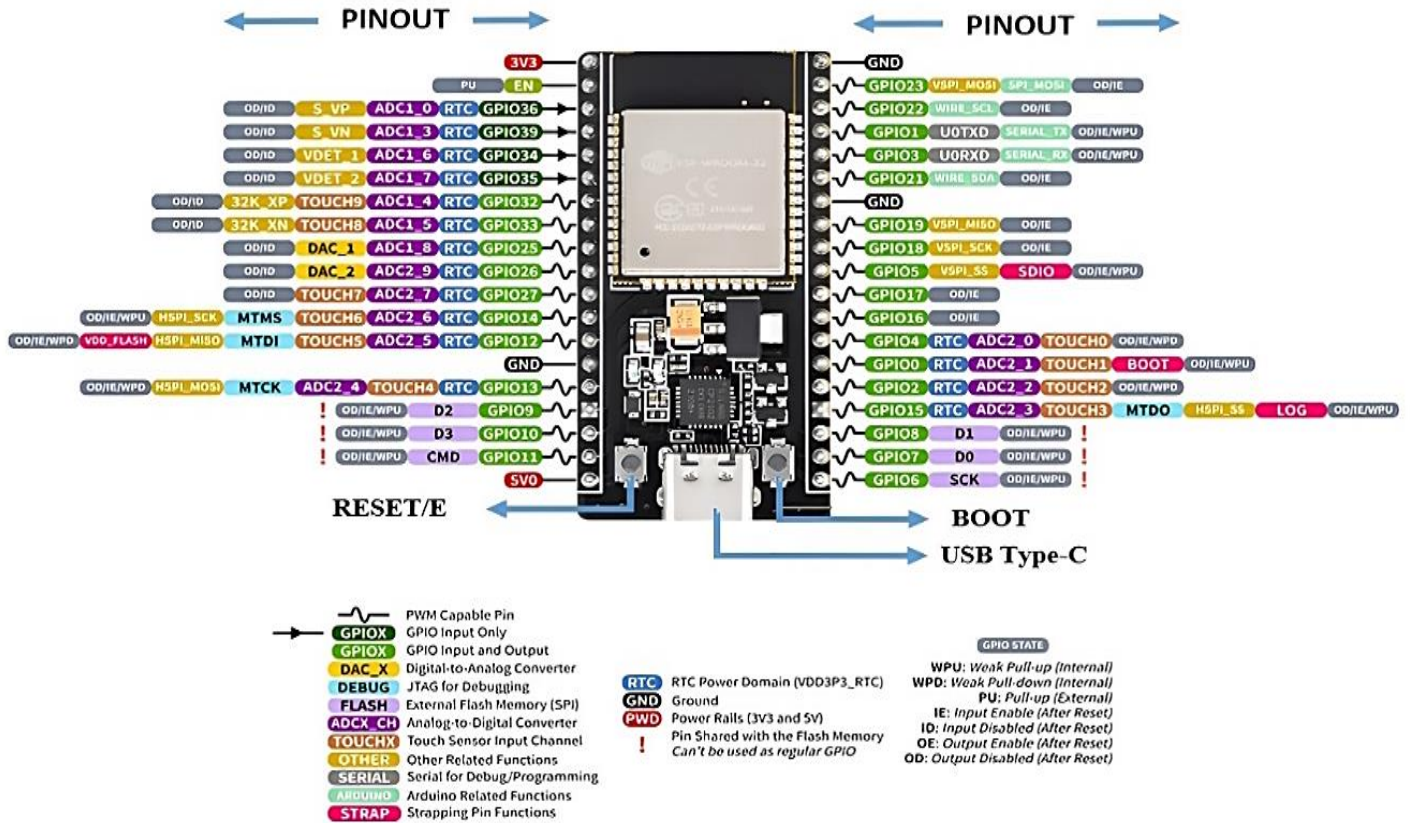


Figure 1. The ESP32 Microcontroller (ESP32 Basics).

2.2.2 Heart Rate and Blood Oxygen Saturation Sensor (MAX30100)

The MAX30100 sensor is plug-and-play and measures blood oxygen saturation levels and heart rate as shown in Fig. 2. It is noninvasive, which means it does not require the introduction of instruments into the patient's body. This sensor features an LED and a photodetector in addition to a high-performance analog front end for accurate measurements (Gade, 2021). Its small size (5.6 mm x 2.8 mm x 1.2 mm) allows it to be used in wearable IoT devices. It communicates via the I2C standard, which makes it easy to integrate with microcontrollers and other embedded devices. It operates at ultra-low power, improving wearable device battery life. It provides accurate readings in any environment due to features including strong motion artifact robustness, ambient light rejection, and a good signal-to-noise ratio (SNR).

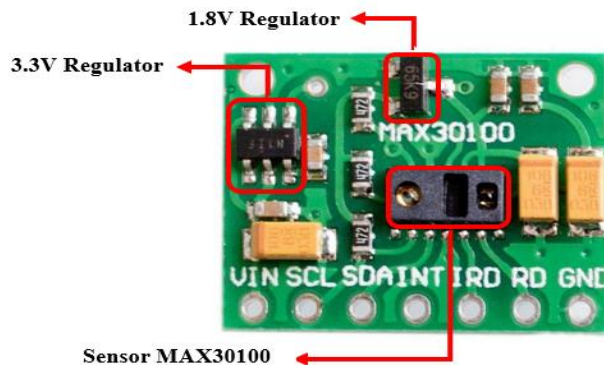


Figure 2. Heart Rate and Blood Oxygen Saturation Sensor (MAX30100).

The high sample rate capability of the MAX30100 enables quick and accurate data collection, which is important in a real-time monitoring system (**Analog, 2009**). An investigation (**Jamal et al., 2023**) of the sensor output data presented in the literature showed that the oxygen saturation data had an accuracy of 98.1% and an appropriately low error rate of 1.90%. Similarly, the heart rate data calibration produced excellent results, with an accuracy of 99.85% and an error rate of 0.15%.

2.2.3 Temperature Sensor (MLX90614)

The MLX90614 infrared thermometer is widely used for body temperature measurement. With the combination of its powerful DSP unit, low-noise amplifier, and 17-bit ADC, the MLX90614 is capable of excellent resolution and accuracy. Its compact size, low cost, and ease of integration are the key features of this sensor. Its temperature range is around (-40°C to +125°C) for sensor temperature and (-70°C to +380°C) for object temperature (**Melexis, 2018**). **Fig. 3** illustrates the MLX90614 sensor.



Figure 3. Temperature Sensor (MLX90614).

2.3 The Software and Programming Language

2.3.1 Node.js

Node.js is "a JavaScript runtime built on Chrome's V8 JavaScript engine" (**Node.js, 2019**), which is written in the C++ programming language. It is based on event-driven; this means that functions are not run sequentially as they would be in code. Where function execution is dependent upon particular events occurring, such as connect, disconnect, timeout, error, etc. The primary benefit of Node.js is its speed compared to alternative approaches. It is designed to process requests asynchronously, in contrast to synchronous languages like Python or Java (**Liang et al., 2017**). As a result, it is appropriate for server-side development. Multiple studies documented the implementation of Node.js on the Internet of Things (**Singh et al., 2019; Mudaliar et al., 2020**).

2.3.2 React.js

React.js is a well-known JavaScript front-end framework for creating interactive and reusable user interface objects. According to the official React.js documentation, it enables the development of complex and large-scale online applications that can manipulate their data without having to refresh the page. It is used to display on the server side by using the Node.js server. Additionally, by converting the source code into static HTML and single-page applications, React.js helps to enhance the presentation of modern web environments.



Compared to conventional data binding, React.js simplifies the boilerplate by using unidirectional dataflow (**Aggarwal, 2018**). Furthermore, the one-page approach leads to improved application performance; for example, an application has only one page instead of multiple pages, which speeds up loading.

2.3.3 JMeter Tool

JMeter is an open-source tool that is used for performance and load testing (**Colantonio, 2022; Tiwari et al., 2023**). It is used to test a system's behavior by simulating thousands of users; JMeter is also used to evaluate IoT systems. It has a simple GUI and displays the results in a tree, table, or chart form that can be saved in XML format. In addition, it provides important features such as extensibility, multithreading, and support for multiple protocols.

2.4 Protocol Types

2.4.1 Message Queue Telemetry Transport (MQTT) Protocol

MQTT is a lightweight, publish-subscribe messaging protocol with low bandwidth and low latency used in IoT applications (**Abdul Ameer et al., 2020; Tsvetanov, 2022; MQTT, 2023**). This protocol is designed for devices with limited power and unreliable networks. These features make it an appropriate choice for web-based data monitoring systems. Every message has its own value and a special topic. This message is transmitted between the publisher and the subscriber via a broker, which is in the middle of MQTT communication, such as Mosquitto. The MQTT message structure is organized using the MQTT specification (**MQTT Specification, 2021**), where the PDU has three fields: fixed header, variable header, and payload. All packets have a fixed header field, which is 2 bytes in size. Additionally, the MQTT packet includes a message type field in the fixed header field indicating the type of connection request being made, as shown in **Table 1. (MQTT, 2023)**.

Table 1. The fixed header format in the MQTT message (**MQTT, 2023**).

bit	7	6	5	4	3	2	1	0
Byte 1	Message Type				DUP flag	QoS Level		RETAIN
Byte 2	Remaining Length							

In the fixed header, the first four bits define message types, while the next four bits of the first byte define header flags, such as DUP (duplicate), quality of service (QoS), and RETAIN. The payload field contains the actual data to be sent and is not present in the MQTT control packet. The MQTT protocol allows for three types of QoS levels that enable different communication. The QoS determines the reliability of the transmission. "QoS 0—At Most Once" does not guarantee message delivery at all; it does not guarantee message receipt by subscribers. Additionally, no acknowledgment packet is transmitted to show the publisher is aware of the message's receipt. "QoS 1—At Least Once" guarantees that a message is delivered at least once, which is a very reliable transmission technique in the MQTT protocol. However, this method is resource-intensive and may result in many messages being sent. "QoS 2—Exactly Once" guarantees that a message is delivered exactly once and ensures delivery to the proper recipient (**Sadeq et al., 2018; Mishra et al., 2020; MQTT, 2023**). The communication is done in the following scenario. First, publishers send the Mosquito Broker their message. The subscriber then receives the broker's message according to the specific topic, as shown in **Fig. 5**.

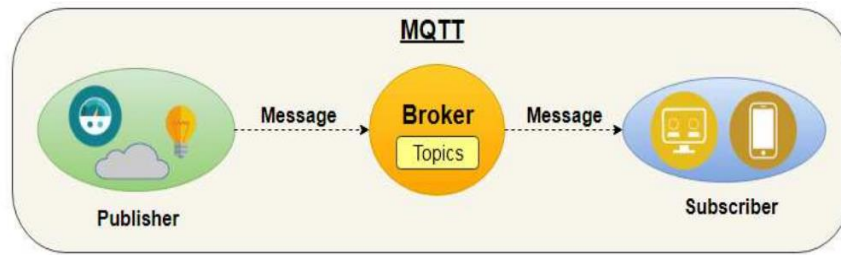


Figure 5. MQTT Communication Scenario (Ansari et al., 2018).

The Mosquitto Broker is an open-source, lightweight broker that supports MQTT versions 5.0, 3.1.1, and 3.1, which is appropriate for a broad variety of devices, including complete servers and low-power single-board computers (Eclipse Mosquitto, 2022). However, it is evident from other studies, such as (Mishra et al., 2021), that a Mosquitto-based solution can process about 20,000 messages per second. Additionally, it includes further functions that protect the IoT system's connectivity. This is to establish a custom configuration for Mosquitto that defines an authentication credential, such as a username and password, to be utilized by each client to log into the broker.

2.4.2 Hypertext Transport Protocol (HTTP)

The application layer protocol known as HTTP was created by Tim Berners-Lee. It is a client/server protocol that uses port 80 as the default port and is based on TCP/IP to deliver data on the World Wide Web. The HTTP protocol is text-based. An HTTP handshake protocol sequence was used to initiate communication. A reliable session is established between the client and server at the start of communication via HTTP. A message request is sent to a server by a client. It contains several parameters, including the message's contents, version, header lines, and uniform resource locators (URL). The server will execute the necessary method and respond to the request after it has been received (Daud et al., 2017). Moreover, this protocol supports a variety of methods of authorization, including Basic, Digest, Bearer token, etc. The general message sequence of the basic authentication method is the same for most authentication schemes, as shown in Fig. 6 (MDN, 2024).

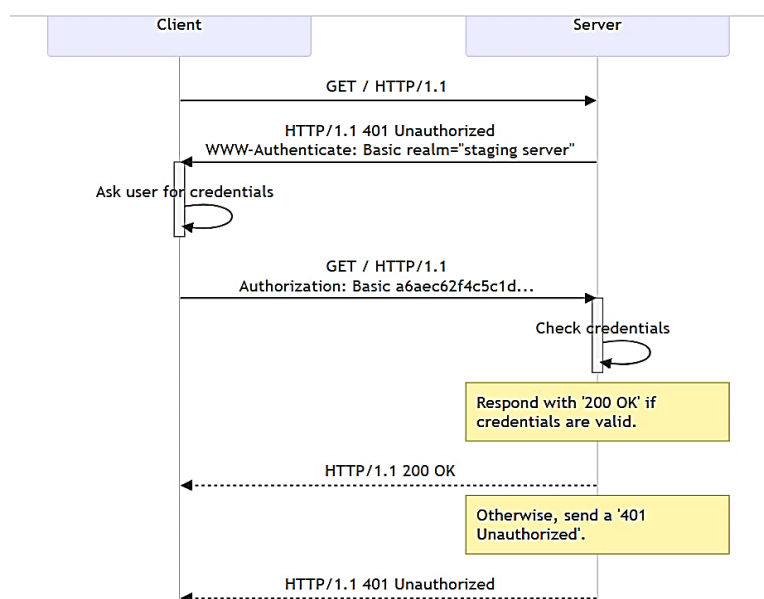


Figure 6. The general message sequence diagram of HTTP authentication (MDN, 2024).

3. DESIGN AND IMPLEMENTATION

3.1 End-to-End System Architecture Design

In this section, the communication flow and end-to-end system architecture are shown in **Fig. 7**, which depicts the components of the overall proposal system. It is composed of three functional layers: a wearable IoT device, MQTT broker, and a web application.

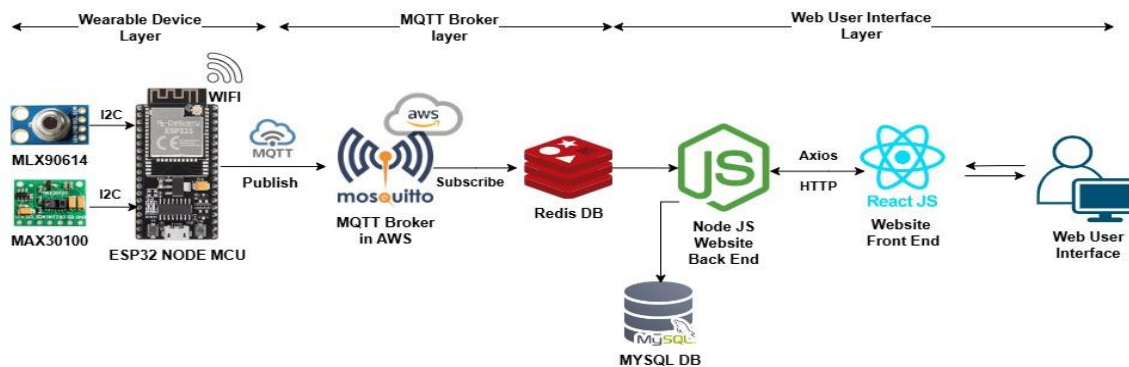


Figure 7. An End-to-end system architecture design.

- **The wearable IoT device layer** consists of heart rate, oxygen level, and temperature sensors that collect physiological signs from the patient's body. These sensors communicate with the ESP32 via the I2C protocol. In addition, the ESP32 is responsible for gathering sensor data, which it subsequently wirelessly transmits to the MQTT broker using the MQTT protocol with a specific topic. In particular, the IoT device, which is most often held by the patient, functions as the MQTT publisher.
- **The MQTT layer** represents the MQTT side in Amazon Web Services (AWS), which consists of the Mosquitto broker and Redis DB. In this layer, a virtual machine running Ubuntu Server OS and a t3. micro instance type is built using Amazon EC2; it has 1 GB of memory and a dual-core. In addition, when the message is published from the IoT device using the MQTT protocol, it goes to the AWS cloud. The message is received by the Mosquitto broker, which then notifies the subscriber of any new sensed data. Finally, the Redis DB serves as a subscriber that stores the received MQTT message.
- **The web application layer** represents the HTTP side in Amazon Web Services (AWS), which consists of the server side (Node.js), the client side (React.js), and MySQL DB. The client side sends an HTTP request to the Pulse Care application server via HTTP to be processed. Then, if the request contains the patient's data, it must be stored in the MySQL database.

3.2 The Architecture Design of the Proposed Pulse Care System

The Pulse Care web application was developed to address the challenges that patients have when they visit a clinic or hospital. This includes the issues of crowding and queuing, paper-based medical records. In addition, the lack of real-time monitoring of patients' vital signs, particularly those with chronic illnesses. Pulse Care is developed based on a JavaScript framework. It is built with Node.js and React.js, which represent the back-end (server-side) and front-end (client-side) of the system, respectively. The application contains a number of web pages depending on the client's permission. It verifies the client role before permitting him, as depicted in **Fig. 8**.

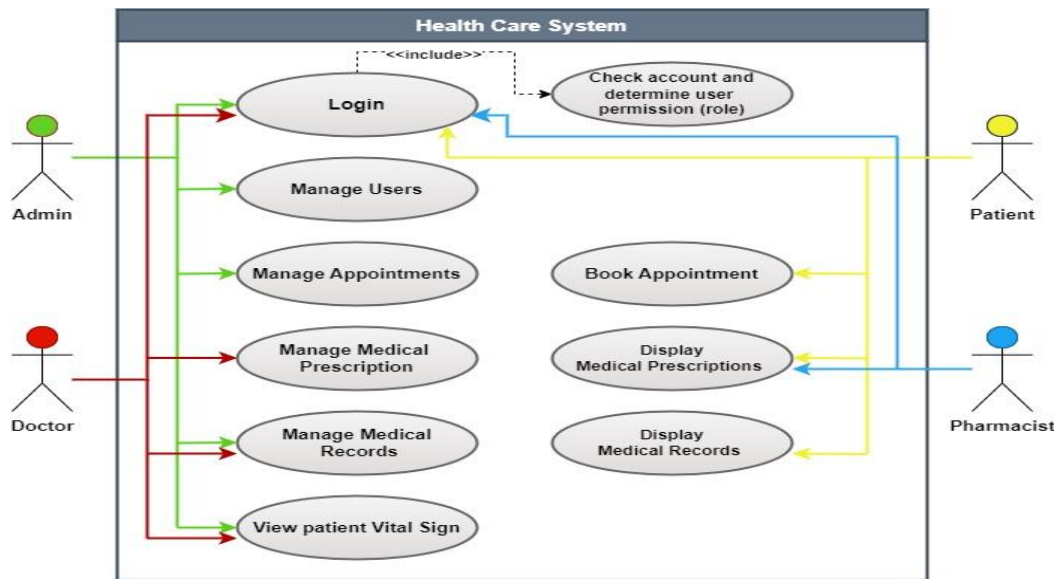


Figure 8. The Pulse Care Application Use Case.

As illustrated in **Fig. 8**, the roles of each system actor are as follows:

- The admin has wider permissions, such as adding a patient's primary information and registering a new member as a doctor, patient, or pharmacist. Additionally, the admin unquestionably has all the other rights in addition to his unique permission to manage appointments .
- The patient's role is to conduct basic activities such as checking prescriptions, booking appointments, and viewing their medical history.
- The doctor can write a prescription, keep track of the patient's medical history, and view vital signs.
- The role of the pharmacist user is to receive prescription alerts and confirm dispensing of them, in addition to the ability to record notes in front of a prescription.

3.3 The Wearable IoT Device Connection and Implementation

The IoT device that the patient is holding, which allows patient vital sign monitoring, is composed of a microprocessor and two sensors. The MAX30100 and MLX90614 sensors provide output through an I2C protocol. These sensors are digital and have four pins: the SDA and SCL to transfer data, another for power, and the final one is for ground. These pins are connected to the ESP32 microcontroller as shown in **Fig. 9**.

This device is powered by a portable power bank with a capacity of 5 volts, which makes the device wearable with its rechargeable power, so it can be worn at any time. This portable power bank connects to the ESP32 and supplies all device parts in addition to the sensors associated with it. Moreover, the Arduino framework and the C++ programming language are utilized to program the ESP32. The MAX30100 library and the Adafruit library are used to connect the ESP32, MAX30100, and MLX90614, respectively. This allows sensors to be integrated, and values derived from real-time measurement data. To enable communication between devices in the system, the AsyncMqttClient software was utilized to connect to the MQTT broker and publish topics (patient ID). MQTT is a lightweight and effective communication protocol for IoT devices, making it an excellent choice for the suggested monitoring system.

In this study, the wearable device functions as the MQTT client (publisher). The data received from the ESP32 is stored by the MQTT broker layer in the Redis database, which provides the information for the chart displayed on the web application.

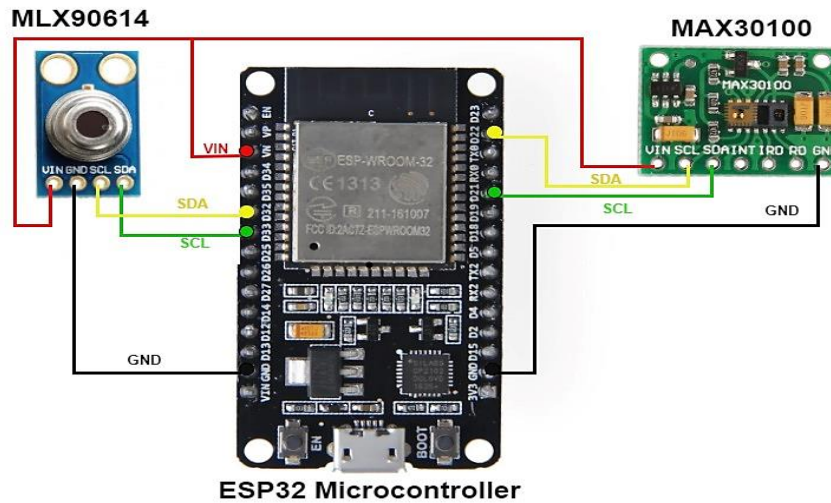


Figure 9. Sensors, Pins, and Connections.

3.4 Implementation of the Pulse Care System

The Pulse Care system consists of four user classifications. First is the administrator, who has access to the entire system. Then the doctor and pharmacist. Finally, the patients and their companions. Depending on the username and password, the application can distinguish between them.

On the login page, as shown in **Fig. 10**, the user can log in to their own page according to the credentials provided. The proposed system's login tables are set up to ensure that, after an administrator creates a new user account, the login information is written to the relevant table together with the user's encrypted password. The goal of encrypting a user's password is for protection, as illustrated in **Fig. 11**.

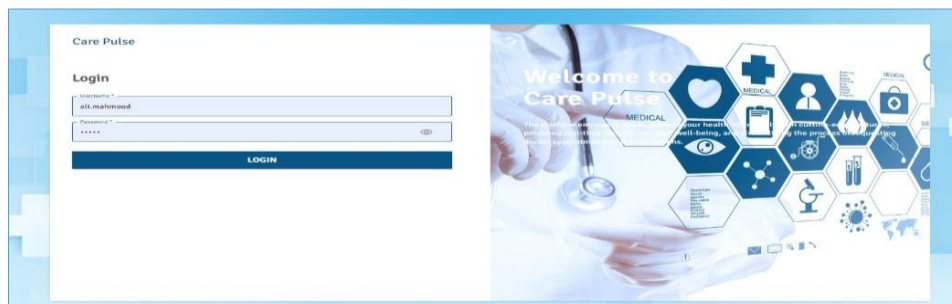


Figure 10. The Login page.

user_id	username	name	email	password
1	xlizer	Mustafa Eltoum	mustafa@gmail.com	\$2b\$10\$FrN0UURlaOQnG5BwpNLqJejK0pLCDi84h24C9f6HEJp...
3	docAdmin	Mohammed Alaa	alaa@mail.com	\$2b\$10\$FrN0UURlaOQnG5BwpNLqJejK0pLCDi84h24C9f6HEJp...
4	pharmacist	Ali Sameer	pharmacist@mail.com	\$2b\$10\$FrN0UURlaOQnG5BwpNLqJejK0pLCDi84h24C9f6HEJp...
5	ali.mahmood	Ali Mahmood	ali.mahmood@mail.com	\$2b\$10\$FrN0UURlaOQnG5BwpNLqJejK0pLCDi84h24C9f6HEJp...
7	ahmed.qassim	Ahmed Qassim Ali		\$2b\$10\$FrN0UURlaOQnG5BwpNLqJejK0pLCDi84h24C9f6HEJp...

Figure 11. The login table.

The system users (physicians, pharmacists, and patients) are registered by the administrator, who is fully authorized to access and register any client. The administrator can select "Add User" from the sidebar. Then insert the user's basic information, such as full name, username, email address, and password. It will be checked to see if the username already exists; an error message is displayed. Otherwise, a successful message will show up if the user is new and has never registered or if his username has never been used, as shown in **Fig. 12**.

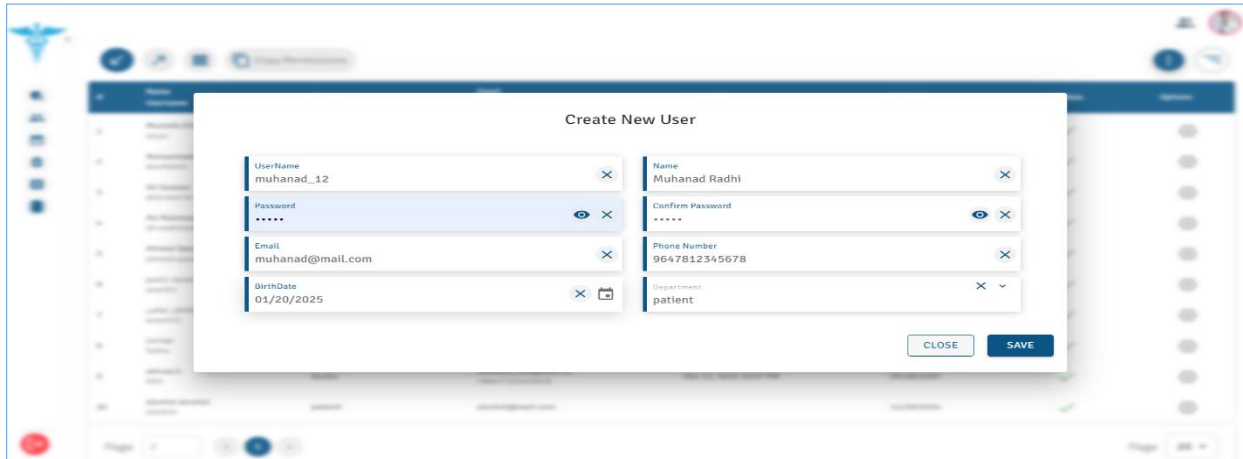
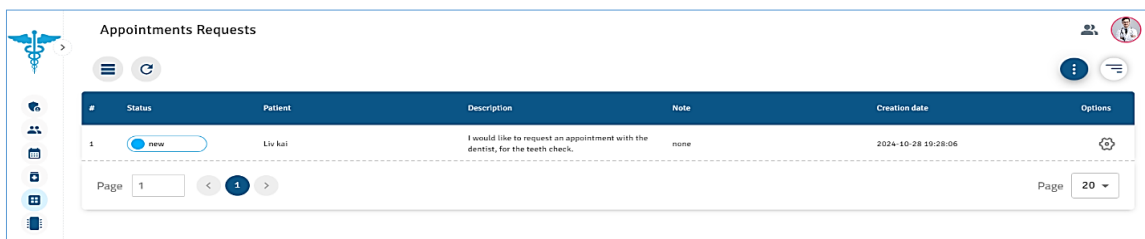


Figure 12. Create a new user page by admin.

Moreover, appointment management is under the responsibility of the admin. Receiving the patient's appointment request is depicted in **Fig. 13**. The administrator can then add the appointment with the appropriate specialist or specific doctor, which includes details of the patient's name and condition as illustrated in **Fig. 14**. After that, send a message to the patient confirming the appointment, which includes the doctor's name and the appointment date, as shown in **Fig. 15**.



#	Status	Patient	Description	Note	Creation date	Options
1	new	Liv kai	I would like to request an appointment with the dentist, for the teeth check.	none	2024-10-28 19:28:06	

Figure 13. Appointment request page.

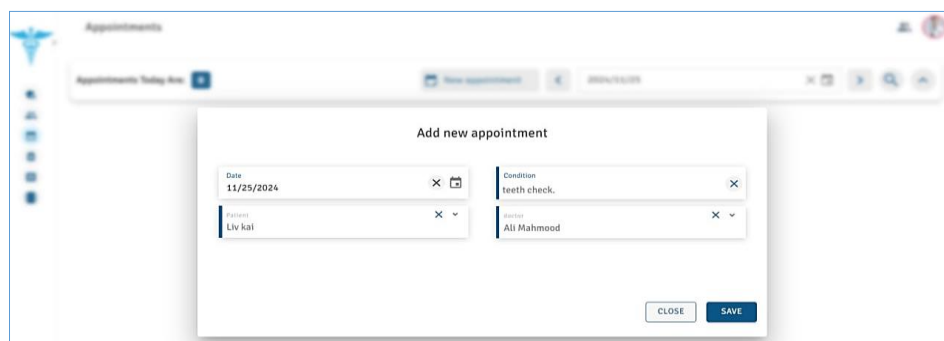


Figure 14. Add new appointment.

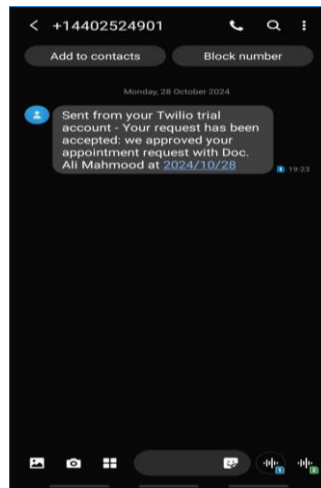


Figure 15. SMS for confirming appointment.

The pharmacist user receives alerts concerning the medications the doctor has prescribed. As well as being able to put notes on prescriptions and verify that the drug is dispensing, as seen on **Fig. 16**.

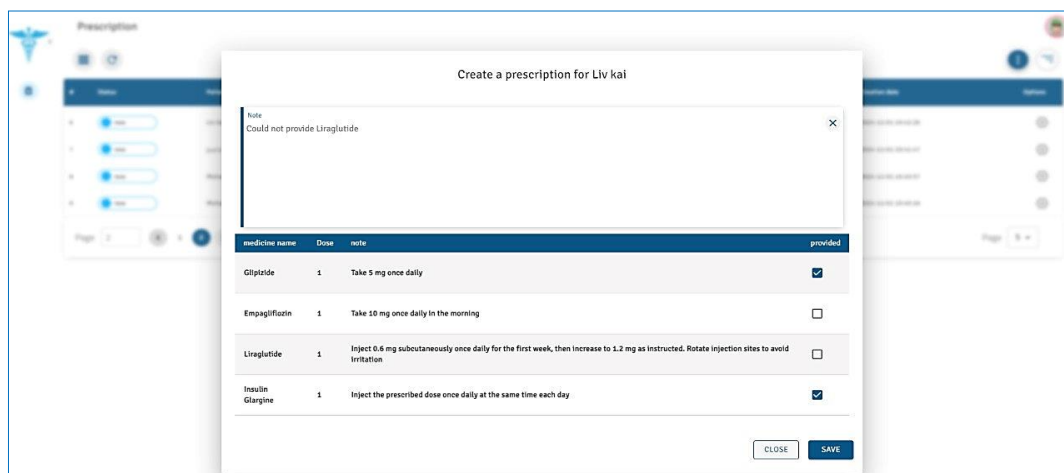


Figure 16. Prescription management system (Pharmacist page).

The EHR page is the most important aspect of this system, which is managed by the doctor, as shown in **Fig. 17**. Additionally, this page displays when a patient logs in to the system to view their personal EHR. It contains four parts. The first part is patient information, which includes personal details such as name, phone number, age, and birth date, as well as any relevant medical data, including blood type, chronic illnesses, previous surgeries, and smoking status. The second part, which is appointments, displays previous and upcoming appointments. The third part, "Patient Status," includes a graph that shows the data produced in real time by the sensors using the patient's IoT wearable device. In addition to displaying historical sensor readings during a specific period. In the last part, past prescriptions are shown along with the option to add a new one. Prescriptions are arranged by date, along with the medical condition and the name of the prescribing doctor.

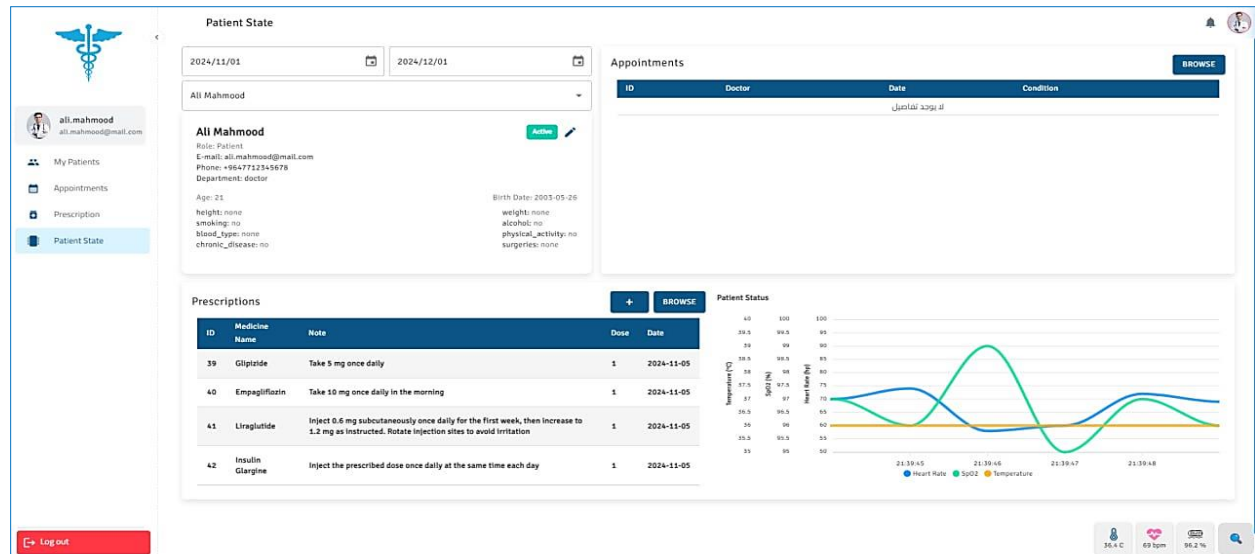


Figure 17. Patient Health Profile (EHR page).

4. RESULTS AND DISCUSSION

This study developed the end-to-end IoT system for enhanced healthcare services. The wearable IoT device was set up correctly, and all of the sensors were linked to the ESP32 microcontroller and a portable power bank for power, as seen in **Fig. 18**. The microcontroller successfully received the data from the sensors and transmitted it in real time to the Mosquitto Broker in AWS.



Figure 18. The wearable device.

Furthermore, the Pulse Care system was implemented using Node.js and React.js, which are based on the JavaScript language. It provides a wide range of functions and involves a number of actors, including the administrator, doctor, patient, and pharmacist. The system implementation has been demonstrated by taking into account the user interface visualization, which is shown in Section 3.4. As compared with related works, the desired features are missing, as shown in **Table 2**. The important contribution of this paper is the combination of these desirable features in the suggested system. As shown in **Table 2**, √ indicates that this feature is achieved, and X indicates that it is not achieved. The comparison demonstrated Pulse Care System's improvements over the current systems. By integrating these functionalities, the system not only improves patient care but also increases operational efficiency within medical practices.

**Table 2.** The comparison with previous related works.

Reference	Real-time monitoring	Used Platform/ Application	Actors	Online Appointment	Standard IoT Protocol (MQTT)	EHR
(Brashdi et al., 2018)	√	Think speak (data visualization)	Doctor	X	X	X
(Durán-Vega et al., 2019)	√	Mobile app	Doctor Patient's Family	X	X	√
(Valsalan et al., 2020)	√	Mobile app (data visualization)	Doctor Patient	X	X	X
(Joseph et al., 2020)	X	Web app	Doctor Nurse Patient	X	X	√
(Ruman et al., 2020)	√	Think speak (data visualization)	Doctor	X	X	X
(Mohammed et al., 2023)	√	Mobile app (data visualization)	Doctor Patient	X	X	X
(Sangeetha akshmi et al., 2023)	√	Think Speak (data visualization)	Doctor	X	X	X
Pulse Care System	√	Web app	Admin Doctor Pharmacist Patient	√	√	√

Moreover, in order to achieve good performance of the Pulse Care system, this paper works on separating the IoT device data from the application data. Then transfer each of them to two different protocols. Starting with IoT device data that is published via the standard IoT protocol (MQTT) to the Mosquitto broker on the AWS EC2 cloud. After the data reaches the broker, the MQTT server subscribes and locates it in the Redis database to be saved. On the other hand, the web application used the HTTP protocol to transfer between the backend and frontend, in addition to inserting client information in the backend MySQL DB. This integration of two types of databases ensures the stability of the system. Redis is an in-memory database, making it fast and suitable for storing dynamic data; therefore, it relieves the strain on the backend database, which stores static data.

The system's performance has been estimated by testing at various levels of QoS. Using Apache JMeter, a semi-realistic setting with a growing patient population was simulated. The findings from an evaluation conducted on 60-second slots at various times of the day are shown in **Table 3**.

Table 3. The average latency through QoS levels 1 and 2.

Number of Users	Avg. Latency QoS0 (ms)	Avg. Latency QoS1 (ms)	Avg. Latency QoS2 (ms)	# Request per min
1	0	101	204	12
10	0	105	218	120
100	0	118	220	1200
200	0	122	247	2400
300	0	142	278	3600



The MQTT broker received 12 requests per minute from each wearable device, with data being transmitted every 5 seconds. The number of requests rises in parallel with the number of devices (patients), reaching 3600 requests per minute with 300 users simultaneously. In addition, it was obvious that as the number of patients rises, the average latency gradually increases at QoS levels 1 and 2, which adds overload to the broker. With a single patient, the average latency of QoS1 and QoS2 is 101 ms and 204 ms, respectively. Additionally, at 300 patients, it reaches 142 ms at QoS1 and 278 ms at QoS2. As a result, QoS 1 uses a lot of resources and may send the same message more than once. But compared to QoS 2, it consumes less latency. Due to the requirement for double confirmation of message delivery at the QoS 2 level, latency is increased. Therefore, QoS 2 might not be the best option for applications that require fast response and real-time performance. Moreover, there is no assurance that subscribers who have subscribed to the topic will receive the message in QoS 0; in fact, no acknowledgment packet confirming the publisher's knowledge of the message's reception is transmitted. Therefore, Apache JMeter considers the message publishing process complete immediately upon sending. For this reason, the average latency value for QoS is zero, due to the nature of QoS0 and the limitations associated with the tool used. The results of average latency through different QoS levels are shown on **Fig. 19**.

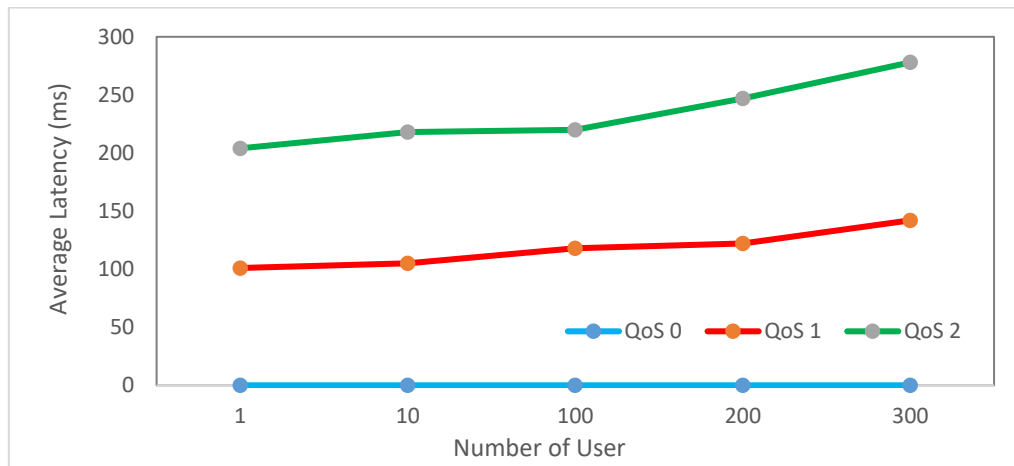


Figure 19. The average latency through QoS Levels 0, 1, 2.

As mentioned before, the sense values are transmitted to the ESP32 microcontroller every five seconds from the wearable device. Every minute, the broker receives about 12 MQTT packets. With each MQTT packet containing approximately 32 bytes of payload, the data rate is about 3.07 kbps. Layer 2 and TCP/IP overheads must be added. However, this data transmission rate is regarded as low and appropriate for transferring small amounts of data periodically using any wireless technology.

5. CONCLUSIONS

The primary goal of the current study is to design and develop an integrated end-to-end system that is intended to improve healthcare services. One of the presented goals is a proof-of-concept solution that is developed to monitor patient vital signs in real time via wearable IoT technology using the MQTT protocol, which simplifies the lives of those with health issues who need to frequently attend the doctor. The other goal is the development of the



Pulse Care system, which facilitates communication between physicians, pharmacists, and patients through simple graphical user interfaces. It has important features, such as patient status monitoring, smart appointments to cut down on hospital queues, and electronic prescription records. In addition, it provides the EHR system to alleviate the issue of manually searching for or losing patient records. The performance of the MQTT broker was evaluated with respect to average latency at various QoS levels as the number of simultaneous users rose. The experiment's findings highlight how crucial it is to select the right quality of service level in order to balance transmission reliability and performance according to the needs of the application. The suggested method has a data rate of roughly 3.07 kbps, which is considered low and suitable for periodically sending small amounts of data via any wireless technology. One limitation of this study is related to the measurement of latency under QoS 0. The latency values were recorded as zero due to the nature of the protocol, which does not require acknowledgments, and the behavior of the JMeter tool that marks the publish action as complete when it is sent. As a result, the reported values under QoS 0 do not accurately reflect end-to-end latency. For this limitation, acknowledge and recommend the use of more accurate measurement techniques in future work, such as ping-pong messaging. The suggested approach can be enhanced in the future by including predictive analytics, which uses patient data to predict the probability of contracting specific diseases. Additionally, adding features like a clinical inventory module and a billing system that will be utilized to determine the cost of the medications that are ordered. Furthermore, more health metrics such as blood pressure, respiration rate, urine output, ECG, and others can be added to the system.

Credit Authorship Contribution Statement

Shahad Ali Ridha: Writing – original draft, Software, Methodology. Mohammed Issam Younis: Writing – review & editing, Software, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

- Abdul Ameer, H.R. and Hasan, H.M., 2020. Enhanced the MQTT protocol by a smart gateway. *Iraqi Journal of Computers, Communications, Control and Systems Engineering*, 20(1), pp. 53-67. <https://doi.org/10.33103/uot.ijccce.20.1.6>.
- Alattar, A.E., and Mohsen, S.A, 2023. Survey on smart wearable devices for healthcare applications. *Wireless Personal Communications*. 132, pp. 775–783. <https://doi.org/10.1007/s11277-023-10639-2>.
- Al Brashdi, Z.B.S., Hussain, S.M., Yosof, K.M., Hussain, S.A., and Singh, A.V., 2018, August. IoT based health monitoring system for critical patients and communication through think speak cloud platform. In *2018 IEEE 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 652-658. IEEE. <https://doi.org/10.1109/ICRITO.2018.8748751>.



- Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M., 2015. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys and Tutorials*, 17(4), pp.2347-2376. <https://doi.org/10.1109/COMST.2015.2444095>.
- Aggarwal, S., 2018. Modern web development using ReactJS. *International Journal of Recent Research Aspects*, 5(1), pp. 133-137.
- Analog, 2009. Pulse oximeter and heart-rate sensor IC for wearable health.
- Ansari, D.B., Rehman, A.U., and Ali, R., 2018. Internet of things (IoT) protocols: A brief exploration of mqtt and coap. *International Journal of Computer Applications*, 179(27), pp. 9-14. <http://dx.doi.org/10.5120/ijca2018916438>.
- Babiuch, M. and Foltýnek, P., 2021. May. Creating a Mobile Application with the ESP32 Azure IoT Development Board Using a Cloud Platform. In 2021, *IEEE 22nd International Carpathian Control Conference (ICCC)*, pp. 1-4. IEEE. <https://doi.org/10.1109/ICCC51557.2021.9454607>.
- Bae, D.H., Jo, I., Choi, Y.A., Hwang, J.Y., Cho, S., Lee, D.G., and Jeong, J., 2018. June. 2B-SSD: The case for dual, byte-and block-addressable solid-state drives. In 2018 *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 425-438. IEEE. <https://doi.org/10.1109/ISCA.2018.00043>.
- Carducci, C.G.C., Monti, A., Schraven, M.H., Schumacher, M., and Mueller, D., 2019. June. Enabling ESP32-based IoT applications in building automation systems. In 2019 *II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*, pp. 306-311. IEEE. <https://doi.org/10.1109/METROI4.2019.8792852>.
- Circuit Basics. Basics of the I2C communication protocol. <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>.
- Colantonio J., Test Guid, 2022. 16 Top Load Testing Software Tools for 2024 (Open-Source Guide). Available at: <https://testguild.com/load-testing-tools/>.
- Daud, M.A. and Suhaili, W.S.H., 2017. Internet of Things (IoT) with CoAP and HTTP protocol: A study on which protocol suits IoT in terms of performance. In *Computational Intelligence in Information Systems: Proceedings of the Computational Intelligence in Information Systems Conference (CIIS 2016)*, pp. 165-174. https://doi.org/10.1007/978-3-319-48517-1_15.
- Droma, F., Bulyaba, H., Ssebawato, J., Nakawooya, K., Musah, K.C., Ongoro, D.A., Suuna, C. and Ndege, R., 2009. An automated system for patient record management: a case study of St. Francis Hospital Nsambya. Bachelor's thesis, Department of Information Technology, Faculty of Computing and Information Technology, Makerere University, Uganda.
- Durán-Vega, L.A., Santana-Mancilla, P.C., Buenrostro-Mariscal, R., Contreras-Castillo, J., Anido-Rifón, L.E., García-Ruiz, M.A., Montesinos-López, O.A., and Estrada-González, F., 2019. An IoT system for remote health monitoring in elderly adults through a wearable device and mobile application. *Geriatrics*, 4(2), P. 34. <https://doi.org/10.3390/geriatrics4020034>.
- Eclipse Mosquitto, 2021. Available at: <https://mosquitto.org/>.
- ESP32 Basics. Available at: <https://lastminuteengineers.com/esp32-pinout-reference/>



Espressif Systems, 2021. ESP32 Wi-Fi & Bluetooth MCU. Available at: <https://www.espressif.com/en/products/socs/esp32>.

Elliott, M. and Coventry, A., 2012. Critical care: the eight vital signs of patient monitoring. *British Journal of Nursing*, 21(10), pp. 621-625. <https://doi.org/10.12968/bjon.2012.21.10.621>.

F. Alshehri and G. Muhammad, 2021. A comprehensive survey of the internet of things (IoT) and AI-based smart healthcare, *IEEE Access*, 9, pp. 3660-3678. <https://doi.org/10.1109/ACCESS.2020.3047960>.

Gade, P., 2021. IoT-based pulse oximeter using ESP8266. Available at SSRN 3918115. <https://dx.doi.org/10.2139/ssrn.3918115>.

Hegde, R., Ranjana, S., and Divya, C.D., 2021, April. Survey on the development of smart healthcare monitoring systems in IoT environment. In *2021, IEEE 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 395-399. IEEE. <https://doi.org/10.1109/ICCMC51019.2021.9418405>.

Islam, M.M., Rahaman, A. and Islam, M.R., 2020. Development of smart healthcare monitoring system in an IoT environment. *SN computer science*, 1, pp. 1-11. <https://doi.org/10.1007/s42979-020-00195-y>.

Jamal, B., Alsaedi, M., and Parandkar, P., 2023. Portable smart emergency system using Internet of Things (IOT). *Mesopotamian Journal of Big Data*, pp. 75-80. <https://doi.org/10.58496/MJBD/2023/011>.

Joseph, B., Gadzama, W.A., and Agu, E.O., 2020. Design and implementation of a secure web-based medical record management system: A case study of Federal University WUKARI (FUW) CLINIC. *International Journal of Computer Applications*, 177(41), pp. 27-33. <https://doi.org/10.5120/ijca2020919908>.

Kadhim, D.J. and Hamad, O. A., 2023. Improving IoT applications using a proposed routing protocol. *Journal of Engineering*, 20(11), pp. 50-62. <https://doi.org/10.31026/j.eng.2014.11.04>.

Khan, M.A., Din, I.U., Kim, B.S., and Almogren, A., 2023. Visualization of a remote patient monitoring system based on the Internet of Medical Things. *Sustainability*, 15(10), P. 8120. <https://doi.org/10.3390/su15108120>.

Kumar, P. and Kumar, A., 2022. Health monitoring system using a microcontroller. *International Journal for Research in Applied Science and Engineering Technology*, 10(6), pp. 607-612. <https://doi.org/10.22214/ijraset.2022.43560>.

Liang, L., Zhu, L., Shang, W., Feng, D., and Xiao, Z., 2017, May. Express supervision system based on NodeJS and MongoDB. In *2017, IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pp. 607-612. IEEE. <https://doi.org/10.1109/ICIS.2017.7960064>.

Li, C., Wang, J., Wang, S., and Zhang, Y., 2024. A review of IoT applications in healthcare. *Neurocomputing*, P. 127017. <https://doi.org/10.1016/j.neucom.2023.127017>.

Mahmood, B.M.R., Younis, M.I., and Ali, H.M., 2023. Construction of a general-purpose infrastructure for RFID-based applications. *Journal of Engineering*, 19(11), pp. 1425-1441. <https://doi.org/10.31026/j.eng.2013.11.06>.



Makeability Lab. Introduction to ESP32. Available at: <https://makeabilitylab.github.io/physcomp/esp32/esp32.html#esp32-pin-list>.

Melexis, 2018. Digital Non-Contact Infrared Thermometer (MLX90614). Available at: <https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can>

Mohammed, B.G. and Hasan, D.S., 2023. Smart healthcare monitoring system using IoT. *Int. J. Interact. Mob. Technol.*, 17(1), pp. 141-152. <https://doi.org/10.3991/ijim.v17i01.34675>.

Mohammed, H.J., 2024. IoT-based low-cost smart health monitoring system using Raspberry Pi Pico W and Blynk application. *Journal of Engineering*, 30(07), pp. 90-108. <https://doi.org/10.31026/j.eng.2024.07.06>.

MQTT, 2023. MQTT - The Standard for IoT Messaging. Available at: <https://mqtt.org/>.

MQTT Specification, 2021. Available at: <https://mqtt.org/mqtt-specification/>.

MDN web document- HTTP authentication, 2024. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Authentication>.

Mishra, B. and Kertesz, A., 2020. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8, pp. 201071-201086. <https://doi.org/10.1109/ACCESS.2020.3035849>.

Mishra, B., Mishra, B. and Kertesz, A., 2021. Stress-testing MQTT brokers: A comparative analysis of performance measurements. *Energies*, 14(18), P. 5817. <https://doi.org/10.3390/en14185817>.

Mostafa, B.S., Miry, A.H., and Salman, T., 2020. Healthcare Monitoring and analytics system based Internet of Things. *Iraqi Journal for Electrical and Electronic Engineering*, pp. 30-36. <https://doi.org/10.37917/ijeee.sceer.3rd.5>.

Mudaliar, M.D. and Sivakumar, N., 2020. IoT IoT-based real-time energy monitoring system using Raspberry Pi. *Internet of Things*, 12, P. 100292. <https://doi.org/10.1016/j.iot.2020.100292>.

Naik, N., 2017, October. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP, and HTTP. In *2017, IEEE International Systems Engineering Symposium (ISSE)*, pp. 1-7. IEEE. <https://doi.org/10.1109/SysEng.2017.8088251>.

Node.js, 2019. Available at: <https://nodejs.org/en/>.

Qadir, G.A. and Hussan, B.K., 2023. An authentication and access control model for healthcare-based cloud services. *Journal of Engineering*, 29(3), pp. 15-26. <https://doi.org/10.31026/j.eng.2023.03.02>.

Ruman, M.R., Barua, A., Rahman, W., Jahan, K.R., Roni, M.J., and Rahman, M.F., 2020, February. IoT-based emergency health monitoring system. In *2020 IEEE International Conference on Industry 4.0 Technology (I4Tech)*, pp. 159-162. IEEE. <https://doi.org/10.1109/I4Tech48345.2020.9102647>.

Sadeq, A.S., Hassan, R., Al-rawi, S.S., Jubair, A.M., and Aman, A.H.M., 2019, September. A QoS approach for Internet of Things (IoT) environment using MQTT protocol. In *2019, IEEE International Conference on Cybersecurity (ICoCSec)*, pp. 59-63. IEEE. <https://doi.org/10.1109/ICoCSec47621.2019.8971097>.

Santoso, D. and Setiaji, F.D., 2015. Non-contact portable infrared thermometer for rapid influenza screening. *2015 International Conference on Automation, Cognitive Science, Optics, Micro Electro-*



Mechanical System, and Information Technology (ICACOMIT), pp. 18-23.
<https://doi.org/10.1109/ICACOMIT.2015.7440147>.

Sangeethalakshmi, K., Preethi, U., and Pavithra, S., 2023. Patient health monitoring system using IoT. *Materials Today: Proceedings*, 80, pp. 2228-2231. <https://doi.org/10.1016/j.matpr.2021.06.188>.

S. Balaji, K. Nathani, and R. Santhakumar, 2019. IoT technology, applications and challenges: A contemporary survey, *Wirel. Pers. Commun.*, vol. 108, no. 1, pp. 363-388.
<https://doi.org/10.1007/s11277-019-06407-w>.

Singh, H.K., Verma, S., Pal, S., and Pandey, K., 2019, August. A step towards Home Automation using IOT. In *2019, IEEE Twelfth International Conference on Contemporary Computing (IC3)*, pp. 1-5. IEEE.
<https://doi.org/10.1109/IC3.2019.8844945>.

Tiwari, V., Upadhyay, S., Goswami, J.K., and Agrawal, S., 2023, April. Analytical evaluation of web performance testing tools: Apache JMeter and SoapUI. In *2023, IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 519-523. IEEE.
<https://doi.org/10.1109/CSNT57126.2023.10134699>.

Tsvetanov, F.A. and Pandurski, M.N., 2022. Selection of protocols for integration of sensory data networks in cloud structures. *Int. J. Online Biomed. Eng.*, 18(9), pp. 29-40.

Turnip, A., Kusumandari, D.E., Simbolon, A.I., and Duhita, N., 2020. Bioactive peptide effect on brain activity identified by 2D brain mapping. *Open Agriculture*, 5(1), pp. 879-887.
<https://doi.org/10.1515/opag-2020-0085>.

Younis, M.I., Abdulkareem, H.F. and Ali, H.M., 2015. Construction of a graduation certificate issuing system based on the digital signature technique. *Journal of Engineering*, 21(6), pp. 15-36.
<https://doi.org/10.31026/j.eng.2015.06.02>.

Valsalan, P., Baomar, T.A.B. and Baabood, A.H.O., 2020. IoT-based health monitoring system. *Journal of Critical Reviews*, 7(4), pp.739-743. <https://dx.doi.org/10.31838/jcr.07.04.137>.

تصميم وتنفيذ نظام إنترنت الأشياء الشامل للرعاية الصحية الذكية استنادًا إلى MQTT و Node.js

شهد علي رضا*، محمد عصام يونس

قسم هندسة الحاسوب، كلية الهندسة، جامعة بغداد، بغداد، العراق

الخلاصة

في السنوات الأخيرة، كان هناك نمو هائل في إنترنت الأشياء (IoT) في مختلف القطاعات، وخاصة الرعاية الصحية. واجه قطاع الرعاية الصحية التقليدي العديد من التحديات، مثل ضعف الخدمات الصحية، والازدحام والطوابير في المراكز الطبية، والبحث اليدوي عن سجلات المرضى أو فقدانها؛ وهناك حاجة إلى نظام صحي ذكي للتغلب على هذه التحديات. تقدم هذه الدراسة تصميم وتنفيذ بنية إنترنت الأشياء الشاملة، والتي تم بناؤها من خلال دمج العديد من التقنيات مفتوحة المصدر. يتم تنفيذ نظام Pulse Care باستخدام Node.js و React.js، والتي توفر ميزات مثل السجلات الصحية الإلكترونية (EHR) والمواعيد الذكية وتوثيق الوصفات الطبية ومراقبة الصحة. بالإضافة إلى ذلك، يتواصل العديد من المستخدمين (المسؤول، ومريض، والطبيب، والصيدلي) باستخدام واجهة رسومية سهلة الاستخدام. من حيث الجهاز القابل للارتداء، فإنه يسمح بتصوير جميع البيانات التي تولدها المستشعرات في الوقت الفعلي (معدل ضربات القلب، ودرجة حرارة الجسم، وأكسجين الدم). يُستخدم ESP32 كوحدة تحكم دقيقة تتواصل عبر شبكة Wi-Fi المدمجة مع وسيط MQTT Mosquitto. وتُستخدم بروتوكولات إنترنت الأشياء القياسية، مثل نقل بيانات قوائم انتظار الرسائل (MQTT) وبروتوكول نقل النص التشعبي (HTTP). ويُقيّم أداء الوسيط من حيث متوسط زمن الوصول عند مستويات جودة الخدمة المختلفة، ومع زيادة عدد المستخدمين المتزامنين. وتُظهر نتائج التجربة أهمية اختيار مستوى جودة الخدمة المناسب بناءً على متطلبات التطبيق. علاوة على ذلك، يبلغ معدل بيانات النظام المقترح حوالي 3.07 كيلوبت/ثانية، وهو معدل منخفض ومناسب لإرسال بيانات صغيرة الحجم عبر أي وسيلة لاسلكية.

الكلمات المفتاحية: إنترنت الأشياء، MQTT، السجلات الصحية الإلكترونية (EHR)، المواعيد الذكية، مراقبة الصحة عن بعد.